1 Einleitung

Moderne unternehmensweite IT-Umgebungen unterliegen zahlreichen Herausforderungen von zunehmender Brisanz. Dies wird verursacht durch

- heterogene Systemlandschaften
- unterschiedliche HW-Plattformen
- unterschiedliche Netze und Netzzugriffe (TCP/IP, SNA, HTTP)
- Verflechtung der Anwendungen mit den Unternehmen

Dadurch entwickeln sich Problemfelder, sei es bei Fusionen, durch Kooperationen oder auch nur durch Rationalisierungsmaßnahmen. Die Unternehmen fordern flexible und skalierbare Anwendungen, gleichzeitig soll die Transaktionssicherheit für Prozesse und Daten gewährleistet bleiben, obwohl die Geschäftsprozesse immer komplexer werden. Die wachsende Globalisierung geht selbstverständlich davon aus, dass Anwendungen im 7x24-Stunden-Betrieb laufen und hochverfügbar sind, um beispielsweise Internetzugriffe auf bestehende Anwendungen über Zeitzonen hinweg zu ermöglichen.

Der klassische Anwendungsserver *open*UTM bietet eine Ablaufumgebung, die allen diesen Anforderungen moderner unternehmenskritischer Anwendungen gewachsen ist, denn *open*UTM verbindet alle Standards und Vorteile von Transaktionsmonitor-Middleware und Message Queuing Systemen:

- Konsistenz der Daten und der Verarbeitung
- Hohe Verfügbarkeit der Anwendungen (nicht nur der Hardware)
- Hohen Durchsatz auch bei großen Benutzerzahlen, d.h. höchste Skalierbarkeit
- Flexibilität bezüglich Änderungen und Anpassungen des IT-Systems

*open*UTM ist Teil des umfassenden Angebots von **openSEAS**. Im Rahmen des Produktangebots **openSEAS** (Open Suite for Enterprise Application Servers) nutzen innovative Produkte die ausgereifte Technologie von *open*UTM:

- Web*Transactions* ermöglicht u.a. zusammen mit *open*UTM moderne eBusiness-Applikationen zu erstellen. Bestehende UTM-Anwendungen können unverändert mit Web*Transactions* in das World Wide Web übernommen und in Portale integriert werden.
- Biz*Transactions* ist das openSEAS-Werkzeug zum Erzeugen wiederverwendbarer Business-Objekte. Die Business-Objekte werden aus bestehenden Anwendungen erzeugt (z.B. BS2000/OSD- und OS/390-Anwendungen, Transaktionsanwendungen mit *open*UTM und SAP R/3) und können je nach Bedarf als JavaBeans, .Net-Komponenten oder Web-Services verfügbar gemacht werden.
- BeanConnect ist eine Familie von Adaptern gemäß der Java Connector Architecture (JCA) von Sun und bietet den standardisierten Anschluss von UTM-Anwendungen an J2EE-Applikationsserver, insbesondere an ORACLE AS 10g. ORACLE AS 10g ist der Applikationsserver, der als Bestandteil von openSEAS als Entwicklungs- und Ablaufplattform aller Geschäftsanwendungen, Portale und Web-Seiten im Grid Computing-Maßstab zur Verfügung steht.

1.1 Konzept und Zielgruppen dieses Handbuchs

Dieses Handbuch richtet sich an Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen.

Es enthält alle Informationen, um ein UTM-Anwendungsprogramm im BS2000/OSD zu erzeugen und eine UTM-Anwendung einzusetzen.

Dieses Handbuch gibt Ihnen in den ersten Kapiteln einen Überblick darüber, wie Sie eine UTM-Anwendung strukturieren, binden und welche Dateien zum Betrieb einer Anwendung notwendig sind. Jeweils eigene Kapitel befassen sich mit dem Starten und Beenden einer UTM-Anwendung und mit dem Programmaustausch bei laufender Anwendung. Die Besonderheiten, die Sie beim Betrieb einer UTM-Datenbank-Anwendung beachten müssen, sind zentral in einem gleichlautenden Kapitel zusammengestellt.

Ausführlich wird darauf eingegangen, wie sich Terminal-Benutzer und andere Clients an eine UTM-Anwendung anmelden können.

Zusätzlich gibt es eigene Kapitel über die Tools, die Ihnen für den Betrieb und die Kontrolle einer UTM-Produktivanwendung zur Verfügung stehen.

Kenntnisse des Betriebssystems werden vorausgesetzt.

1.2 Wegweiser durch die Dokumentation zu *open*UTM

In diesem Abschnitt erhalten Sie einen Überblick über die Handbücher zu *open*UTM und zum Produktumfeld von *open*UTM.

1.2.1 *open*UTM-Dokumentation

Die *open*UTM-Dokumentation besteht aus Handbüchern, einer Online-Hilfe für den grafischen Administrationsarbeitsplatz *open*UTM WinAdmin sowie einer Freigabemitteilung für jede Plattform, auf der *open*UTM freigegeben wird.

Es gibt Handbücher, die für alle Plattformen gültig sind, sowie Handbücher, die jeweils für BS2000/OSD bzw. für UNIX, Linux und Windows gelten.

Sämtliche Handbücher sind als PDF-Datei im Internet verfügbar unter der Adresse
`http://manuals.fujitsu-siemens.com/servers/man/man_de/utm_man.htm`.
Die meisten Handbücher können außerdem auch als gebundene Papierexemplare bestellt werden. Zusätzlich gibt es eine CD-ROM, die die komplette *open*UTM-Dokumentation enthält.

Die folgenden Abschnitte geben einen aufgabenbezogenen Überblick über die Dokumentation zu *open*UTM V5.2. Eine vollständige Liste der Dokumentation zu *open*UTM finden Sie im Literaturverzeichnis auf Seite 315.

Einführung und Überblick

Das Handbuch **Konzepte und Funktionen** gibt einen zusammenhängenden Überblick über die wesentlichen Funktionen, Leistungen und Einsatzmöglichkeiten von *open*UTM. Es enthält alle Informationen, die Sie zum Planen des UTM-Einsatzes und zum Design einer UTM-Anwendung benötigen. Sie erfahren, was *open*UTM ist, wie man mit *open*UTM arbeitet und wie *open*UTM in die Betriebssysteme BS2000/OSD, UNIX, Linux und Windows eingebettet ist.

Programmieren

- Zum Erstellen von Server-Anwendungen über die KDCS-Schnittstelle benötigen Sie das Handbuch **Anwendungen programmieren mit KDCS für COBOL, C und C++**, in dem die KDCS-Schnittstelle in der für COBOL, C und C++ gültigen Form beschrieben ist. Diese Schnittstelle umfasst sowohl die Basisfunktionen des universellen Transaktionsmonitors als auch die Aufrufe für verteilte Verarbeitung. Es wird auch die Zusammenarbeit mit Datenbanken beschrieben.

- Wollen Sie die X/Open-Schnittstellen nutzen, benötigen Sie das Handbuch **Anwendungen erstellen mit X/Open-Schnittstellen**. Es enthält die UTM-spezifischen Ergänzungen zu den X/Open-Programmschnittstellen TX, CPI-C und XATMI sowie Hinweise zu Konfiguration und Betrieb von UTM-Anwendungen, die X/Open-Schnittstellen nutzen. Ergänzend dazu benötigen Sie die X/Open-CAE-Specification für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument ***open*UTM Data Marshalling mit XML**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden. Diese Beschreibung ist nur als PDF-Dokument erhältlich (online und auf CD).
- Für BS2000/OSD gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1. Diese sind nur als PDF-Datei verfügbar (online und auf CD).

Konfigurieren

Zur Definition von Konfigurationen steht Ihnen das Handbuch **Anwendungen generieren** zur Verfügung. Darin ist beschrieben, wie Sie für eine UTM-Anwendung mit Hilfe des UTM-Tools KDCDEF die Konfiguration definieren und die KDCFILE erzeugen. Zusätzlich wird gezeigt, wie Sie wichtige Verwaltungs- und Benutzerdaten mit Hilfe des Tools KDCUPD in eine neue KDCFILE übertragen, z.B. beim Umstieg auf eine neue *open*UTM-Version oder nach Änderungen in der Konfiguration.

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000/OSD bzw. UNIX/Windows) das Handbuch **Einsatz von *open*UTM-Anwendungen**. Dort ist beschrieben, wie man ein UTM-Anwendungsprogramm bindet und startet, wie man sich bei einer UTM-Anwendung an- und abmeldet und wie man Anwendungsprogramme strukturiert und im laufenden Betrieb austauscht. Außerdem enthält es die UTM-Kommandos, die dem Terminal-Benutzer zur Verfügung stehen.

Administrieren und Konfiguration dynamisch ändern

- Für das Administrieren von Anwendungen finden Sie die Beschreibung der Programmschnittstelle zur Administration und die UTM-Administrationskommandos im Handbuch **Anwendungen administrieren**. Es informiert über die Erstellung eigener Administrationsprogramme und über die Möglichkeiten der zentralen Administration mehrerer Anwendungen. Darüber hinaus beschreibt es, wie Sie Message Queues und Drucker mit Hilfe der KDCS-Aufrufe DADM und PADM administrieren können.

- Wenn Sie den grafischen Administrationsarbeitsplatz ***open*UTM WinAdmin** einsetzen, dann steht Ihnen folgende Dokumentation zur Verfügung:
  - Die **WinAdmin-Beschreibung** bietet einen umfassenden Überblick über den Funktionsumfang von WinAdmin und das Handling von WinAdmin. Dieses Dokument wird mit der Software ausgeliefert und ist zusätzlich auch online als PDF-Datei verfügbar.
  - Das **Online-Hilfesystem** beschreibt kontextsensitiv alle Dialogfelder und die zugehörigen Parameter, die die grafische Oberfläche bietet. Außerdem wird dargestellt, wie man WinAdmin konfiguriert, um die UTM-Anwendungen administrieren und generieren zu können.

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für die Betriebssysteme UNIX/Windows und für das Betriebssystem BS2000/OSD). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das Verhalten im Fehlerfall, das Meldungswesen von *open*UTM, sowie alle von *open*UTM ausgegebenen Meldungen und Returncodes.

***open*UTM-Clients erstellen**

Wenn Sie Client-Anwendungen für die Kommunikation mit UTM-Anwendungen erstellen wollen, stehen Ihnen folgende Handbücher zur Verfügung:

- Das Handbuch ***open*UTM-Client für Trägersystem UPIC** beschreibt Erstellung und Einsatz von Client-Anwendungen, die auf UPIC basieren. Neben der Beschreibung der Schnittstellen CPI-C und XATMI erhalten Sie Informationen, wie Sie die C++-Klassen oder ActiveX für die schnelle und einfache Programmerstellung nutzen können.
- Das Handbuch ***open*UTM-Client für Trägersystem OpenCPIC** beschreibt, wie man OpenCPIC installiert und konfiguriert. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.
- Für die mit BeanConnect ausgelieferten **JUpic-Java-Klassen** wird die Dokumentation mit der Software ausgeliefert. Diese Dokumentation besteht aus Word- und PDF-Dateien, die die Einführung und die Installation beschreiben, sowie aus einer Java-Dokumentation mit der Beschreibung der Java-Klassen.

Kopplung mit der IBM-Welt

Wenn Sie aus Ihrer UTM-Anwendung mit Transaktionssystemen von IBM kommunizieren wollen, benötigen Sie außerdem das Handbuch **Verteilte Transaktionsverarbeitung zwischen *open*UTM und CICS-, IMS- und LU6.2-Anwendungen**. Es beschreibt die CICS-Kommandos, IMS-Makros und UTM-Aufrufe, die für die Kopplung von UTM-Anwendungen mit CICS- und IMS-Anwendungen benötigt werden. Die Kopplungsmöglichkeiten werden anhand ausführlicher Konfigurations- und Generierungsbeispiele erläutert. Außerdem beschreibt es die Kommunikation über *open*UTM-LU62, sowie dessen Installation, Generierung und Administration.

1.2.2 Dokumentation zum openSEAS-Produktumfeld

Die Verbindung von *open*UTM zum openSEAS-Produktumfeld wird im *open*UTM-Handbuch **Konzepte und Funktionen** kurz dargestellt. Die folgenden Abschnitte zeigen, welche der openSEAS-Dokumentationen für *open*UTM von Bedeutung sind.

Web-Anbindung

Zum Anschließen neuer und bestehender UTM-Anwendungen an das Web mit dem Produkt Web*Transactions* benötigen Sie die folgenden Handbücher:

- Das einführende Handbuch **Web*Transactions* - Konzepte und Funktionen** gibt Ihnen einen Überblick über die Leistungsfähigkeit und Einsatzmöglichkeiten des Produkts Web*Transactions* und informiert Sie über dessen Eigenschaften und Funktionsweisen. Das Handbuch erläutert das Objekt-Konzept und den dynamischen Ablauf einer Web*Transactions*-Sitzung.
- Im Handbuch **Web*Transactions* - Template-Sprache** sind alle Sprachmittel der Template-Sprache WTML erläutert. Es enthält zahlreiche Beispiele, die die Sprachmittel veranschaulichen und Ihnen den Einsatz erleichtern.
- Die Handbücher **Web*Transactions* - Anschluss an *open*UTM-Anwendungen über UPIC** und **Web*Transactions* - Anschluss an OSD-Anwendungen** beschreiben die notwendigen Schritte, um UTM-Dialoganwendungen über die UPIC- bzw. die Terminal-Schnittstelle an das Web anzuschließen. An einem konkreten Beispiel werden diese Schritte nochmals verdeutlicht.

Anwendungsintegration

Wenn Sie UTM-Anwendungen und -Services in Microsoft-Anwendungen integrieren möchten, dann steht Ihnen dafür das Produkt Biz*Transactions* mit dem Handbuch **Biz*Transactions* - Anwendungsintegration mit Business Objekten** zur Verfügung. Es beschreibt neben der Installation und Konfiguration auch die Konzepte und Komponenten von Biz*Transactions*. Außerdem wird die Administration und das Entwickeln von integrierten Client-Anwendungen erläutert.
Für Biz*Transactions* wird mit der Software auch ein Online-Hilfesystem ausgeliefert.

Ablaufumgebung für Enterprise JavaBeans

Die Dokumentation zu BeanConnect ist in Vorbereitung.

Die Handbücher zu ORACLE AS 10g sind bei ORACLE beziehbar.

1.2.3 Readme-Dateien

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. den produktspezifischen Readme-Dateien.

- BS2000/OSD:

  Sie finden die Informationen auf Ihrem BS2000-Rechner in der Freigabemitteilung (Dateiname SYSFGM.UTM.052.D*)* und eventuell zusätzlich in einer Readme-Datei (Dateiname SYSRME.UTM.052.D*)*. Die Benutzerkennung, unter der sich die Datei befindet, erfragen Sie bitte bei Ihrer zuständigen Systembetreuung. Die Datei können Sie mit dem Kommando /SHOW-FILE oder mit einem Editor ansehen oder auf einem Standarddrucker mit folgendem Kommando ausdrucken:

  ```
  /PRINT-DOCUMENT dateiname,LINE-SPACING=*BY-EBCDIC-CONTROL
  ```

1.3 Neuerungen in *open*UTM V5.2

*open*UTM bietet eine Reihe neuer Funktionen auf Server- und auf Client-Seite. Der grafische Administrationsarbeitsplatz WinAdmin wurde ebenfalls funktionell erweitert.

Die folgenden Abschnitte gehen näher auf die Neuerungen in den einzelnen Bereichen ein.

1.3.1 Neue Plattformen

- *open*UTM bietet auf Solaris-Systemen volle Ablauffähigkeit im **64-bit-Modus**. Der Ablaufmodus (32/64-bit) kann über die Administrationsschnittstelle KDCADMI abgefragt werden.
- Der *open*UTM-Server und *open*UTM-Client laufen auch auf den aktuellen Windows-Plattformen Windows XP und Windows 2003.

1.3.2 Neue Server-Funktionen

Erweiterungen beim Message Queuing

- Browse-Funktion

  Mit der neuen Browse-Funktion können Nachrichten von Service-gesteuerten Queues gelesen werden, ohne sie zu löschen. Damit lassen sich Nachrichten einer Queue auch parallel von mehreren Vorgängen lesen.

  Zusammen mit der Browse-Funktion ist es möglich, Nachrichten einer Service-gesteuerten Queue gezielt zu verarbeiten (nicht nur sequenziell).

- Redelivery-Funktion

  Für Nachrichten in Service-gesteuerten Queues und Nachrichten an Hintergrundaufträge ist eine erneute Zustellung möglich, wenn während der Verarbeitung ein Fehler auftritt (abnormales Beenden des Asynchron-Services bzw. Rücksetzen der Transaktion). Die Anzahl der erneuten Zustellungen lässt sich im Programm ermitteln, die maximale Anzahl ist per Generierung einstellbar.

Verschlüsselung nach dem AES/RSA-Verfahren

Die Verschlüsselung ist wahlweise nach dem DES/RSA oder dem AES/RSA-Verfahren möglich. Für den RSA-Schlüssel sind die Modulo-Längen 200, 512, 1024 und 2048 möglich. Für den Zugangsschutz gegenüber Clients sind damit 4 Verschlüsselungsebenen möglich, abhängig von der Länge des RSA-Schlüssels. Beim Zugriffsschutz können 2 Ebenen definiert werden (DES- oder AES-Verschlüsselung).

Alle RSA-Schlüssel können per programmierter Administration (KDCADMI) neu erzeugt, gelesen, aktiviert, deaktiviert oder gelöscht werden.

Erweiterung beim XML Data Marshalling

- Es können Namensräume definiert werden, um in einem XML-Dokument mehrere Namensspezifikationen zu verwenden.
- Es lassen sich mehrere Kindknoten mit gleichem Namen erzeugen. Auf diese kann mit Lese- und Schreibaufrufen zugegriffen werden.
- Daten können umcodiert und Zeichensätze können ermittelt/gesetzt werden.
- Attribute können erzeugt werden, auf die mit Lese-/Schreib- und Positionieraufrufen zugegriffen werden kann.

Sonstige Änderungen

- Erweiterte Transaktionssteuerung:
  Eine Transaktion kann jetzt so beendet oder zurückgesetzt werden, dass der Programmkontext erhalten bleibt und das Programm anschließend fortgesetzt werden kann (neue PGWT-Varianten).
- Die Namen von Teilprogrammen können zusätzliche Sonderzeichen enthalten.
- In BS2000/OSD werden UTM-Systemcode, KDCROOT-Modul und Dienstprogramme über LLMs geladen. Damit sind Korrekturen für einzelne Komponenten mittels Software-Korrekturverfahren möglich (z.B. Austausch ganzer Module).
- In BS2000/OSD ist bei KDCUPD die Übertragung der „Locales“ steuerbar.
- In BS2000/OSD-Teilprogrammen können die Druckoptionen für RSO-Drucker übergeben werden.
- Für BS2000/OSD-Datenbanken kann das Datenbank-Verbindungsmodul aus dem IMON-Installationspfad nachgeladen werden. Dadurch ist die UTM-Anwendung unabhängig von Installationspfaden und Bibliotheksnamen des Datenbanksystems.

1.3.3 Neue Client-Funktionen

- Erweiterung der Sicherheitsfunktionen
  - Der UPIC-Client unterstützt alle Verschlüsselungsebenen des *open*UTM-Servers. Verschlüsselung nach dem DES/RSA bzw. AES/RSA-Verfahren mit Modulo-Länge des Schlüssels von 200, 512, 1024 und 2048.
  - Der UPIC-Client kann das detaillierte Ergebnis der Berechtigungsprüfung in Form eines erweiterten Returncodes ermitteln.
- Neue Funktionen zum Setzen von Conversation Characteristics

  Die Conversation Characteristics CHARACTER_CONVERTION (ASCII-EBCDIC-Konvertierung) und COMMUNICATION_PROTOCOL sind im Programm veränderbar.
- Verbesserte Steuerung des Wiederanlaufs

  Der Client kann einen 8 byte langen Kontext an *open*UTM übergeben und nach einem Wiederanlauf wieder auslesen.
- Abfragen des Server-Status

  Der Client kann den aktuellen Vorgangs- und Transaktionsstatus des Servers abfragen.
- Weiterentwicklung des ActiveX Control

  Das Control wurde um die Properties *PartnerLUName* und *PartnerPort* erweitert.

1.3.4 Erweiterungen für *open*UTM WinAdmin

- Es können benutzerspezifische Aktionen definiert werden wie z.B. das Öffnen von Dateien, Speichern von Listen in Dateien und Aufrufen von Skripts und Programmen.
- Die Statistik-Funktionen wurden erweitert um Schwellwert-Alarm, Erfassung von Statistikwerten einzelner UTM-Objekte, verbesserte Statistik-Grafik und Ausdrucken der Statistik-Grafik.
- Die Bean*Transactions* Management Console kann von WinAdmin aus aufgerufen werden und der Import von Bean*Transactions*-Dateien ist möglich (Container-Properties).
- Die erweiterten Encryption-Funktionen von *open*UTM V5.2 können per WinAdmin administriert werden. Mit WinAdmin kann man RSA-Schlüssel für unterschiedliche Verschlüsselungsebenen erzeugen, lesen, aktivieren/deaktivieren und löschen.
- Anzeige des OSI-LPAP in der Liste der generierten OSI-Associations.

1.4 Darstellungsmittel

Metasyntax

Die in diesem Handbuch verwendete Metasyntax können Sie der folgenden Tabelle entnehmen:

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Konstanten (Namen von Aufrufen, Anweisungen, Feldnamen, Kommandos und Operanden etc.), die in dieser Form anzugeben sind.	`LOAD-MODE=STARTUP`
kleinbuchstaben	In Kleinbuchstaben sind in Syntaxdiagrammen und Operandenbeschreibung die Platzhalter für Operandenwerte dargestellt.	`KDCFILE=filebase`
kleinbuchstaben	Im Fließtext werden Variablen sowie Namen von Datenstrukturen und Feldern in kursiven Kleinbuchstaben dargestellt.	*utmpfad* ist das UTM-Installationsverzeichnis
`Schreibmaschinen-schrift`	In Schreibmaschinenschrift werden im Fließtext Kommandos, Dateinamen, Meldungen und Beispiele ausgezeichnet, die in genau dieser Form eingegeben werden müssen bzw. die genau diesen Namen oder diese Form besitzen.	Der Aufruf `tpcall`
{ } und |	In geschweiften Klammern stehen alternative Angaben, von denen Sie eine auswählen müssen. Die zur Verfügung stehenden Alternativen werden jeweils durch einen Strich getrennt aufgelistet.	`STATUS={ ON | OFF }`
[ ]	In eckigen Klammern stehen wahlfreie Angaben, die entfallen können.	`KDCFILE=( filebase [, { SINGLE| DOUBLE} ] )`
( )	Kann für einen Operanden eine Liste von Parametern angegeben werden, sind diese in runde Klammern einzuschließen und durch Kommata zu trennen. Wird nur ein Parameter angegeben, kann auf die Klammern verzichtet werden.	`KEYS=(key1,key2,...key`*n*`)`
<u>Unterstreichen</u>	Unterstreichen kennzeichnet den Standardwert.	`CONNECT= { A/YES | NO }`

Formale Darstellung	Erläuterung	Beispiel
Kurzform	Die Standardkurzform für Anweisungen, Operanden und Operandenwerte wird „fett“ hervorgehoben. Die Kurzform kann alternativ angegeben werden.	TRANSPORT-SELEKTOR=c'C'
...	Punkte zeigen die Wiederholbarkeit einer syntaktischen Einheit an. Außerdem kennzeichnen die Punkte Ausschnitte aus einem Programm, einer Syntaxbeschreibung o.ä.	KDCDEF starten : : OPTION DATA=statement_file : END

Sonstige Symbole



Alle Stellen, die mit dem nebenstehenden Symbol gekennzeichnet sind, verweisen auf umfassende und detaillierte Informationen zum jeweiligen Thema.



für Hinweistexte.



für Warnhinweise.

2 Struktur des Anwendungsprogramms festlegen

In diesem Kapitel wird auf die Strukturierung eines Anwendungsprogramms eingegangen. Es wird vorausgesetzt, dass Sie zum Binden den BINDER verwenden und dass Sie das Anwendungsprogramm mit BLS laden.

Ein UTM-Anwendungsprogramm kann unterschiedlich strukturiert und auf verschiedene Weise geladen werden. Generell lassen sich fünf verschiedene Bereiche unterscheiden:

- Shareable Programme im Systemspeicher.

  Programmteile, die in den Systemspeicher geladen werden, stehen allen Prozessen eines BS2000/OSD gemeinsam zur Verfügung. Daher sollten Sie in den Systemspeicher in erster Linie Programme laden, die anwendungsunabhängig sind, wie z.B. die shareable Teile von Laufzeitsystemen der Programmiersprachen. Sehen Sie dazu Abschnitt „Shared Code im Systemspeicher“ auf Seite 26.

  Die shareable Module müssen vom Systemadministrator als nicht-privilegiertes Subsystem geladen werden.

- Statisch gebundene Programmteile.

  Die ROOT-Systemmodule sowie die von diesen benötigten Module des Laufzeitsystems (ILCS) und die Meldungsmodule, für die in der Generierung keine Nachladebibliothek angegeben wurde, können Sie statisch in das Start-LLM einbinden. Das von KDCDEF erzeugte und anschließend assemblierte ROOT-Tabellenmodul kann auch beim Start dynamisch nachgeladen werden.

  Auch weitere Teile der Anwendung können Sie in den statischen Teil einbinden, jedoch dürfen diese keine Externverweise auf Module enthalten, die Sie in einen von *open*UTM verwalteten Common Memory Pool laden.

  Den statischen Teil einer UTM-Anwendung laden Sie mit dem Kommando

  START-PROGRAM bzw. LOAD-PROGRAM

- Shareable Programme in Common Memory Pools.

  Programmteile, die von allen Prozessen einer UTM-Anwendung gemeinsam benutzt werden können, wie z.B. die shareablen Teile Ihrer Teilprogramme der UTM-Anwendung oder auch Formate oder Datenbereiche, sollten Sie in Common Memory Pools laden. Sehen Sie dazu Abschnitt „Shared Code in Common Memory Pools“ auf Seite 28.

- Beim Start der Anwendung zu ladende Programmteile.

  Programmteile, die ständig von der UTM-Anwendung benötigt werden oder die Externverweise zu shareable Teilen der Anwendung enthalten, müssen Sie beim Anwendungsstart dynamisch nachladen lassen.

  Auch das vom KDCDEF erzeugte und assemblierte ROOT-Tabellenmodul wird dynamisch nachgeladen, falls Sie es nicht statisch zum Start-LLM dazugebunden haben.

- Beim Programmaufruf zu ladende Programmteile.

  Programmteile, die **nicht** ständig von der UTM-Anwendung benötigt werden, können Sie so generieren, dass diese erst zum Zeitpunkt des ersten Aufrufs nachgeladen werden. Diese Programmteile müssen Sie derart zu LLMs vorbinden, dass alle offenen Externverweise beim Laden aus den bereits im Speicher vorhandenen Modulen befriedigt werden können.

In welchen der Speicherbereiche und zu welchem Zeitpunkt die Programmteile geladen werden, ist abhängig von den Angaben, die Sie bei der Generierung der Anwendung mit KDCDEF machen (siehe *open*UTM-Handbuch „Anwendungen generieren“).



Informationen dazu finden Sie im Handbuch in den folgenden Abschnitten:

- „Binden mit BINDER“ auf Seite 39
- „Beispiel einer UTM-Startprozedur“ auf Seite 92
- Kapitel „Programmaustausch im Betrieb“ auf Seite 147

2.1 Lademodule generieren

Wenn Sie die BLS-Funktionalität nutzen, braucht nur ein Teil der Anwendung statisch zum Anwendungsprogramm gebunden werden (Start-LLM, siehe Abschnitt „Binden mit BINDER“ auf Seite 39). Die anderen Teile des Anwendungsprogramms müssen Sie in Form von dynamisch nachladbaren Lademodulen bereitstellen.

Bei den Lademodulen kann es sich um einzelne Objektmodule (OM) handeln, die in einer Bindemodulbibliothek (OML) oder einer Programmbibliothek (PL) als Elemente vom Typ R bereitgestellt werden, oder um mit dem BINDER erzeugte Bindelademodule (LLM), die in einer Programmbibliothek als Elemente vom Typ L bereitgestellt werden. Gruppen von Objekten können Sie mit dem BINDER zu LLMs vorbinden, siehe Abschnitt „Binden von LLMs“ auf Seite 41.

Bei der KDCDEF-Generierung müssen die einzelnen Lademodule der Anwendung für den Einsatz des BLS mit LOAD-MODULE-Anweisungen generiert werden. Die Zuordnung von Objekten (Teilprogramme und gemeinsam nutzbare Datenbereiche) zu den Lademodulen wird ebenfalls bei der Generierung festgelegt (Operand LOAD-MODULE in den PROGRAM- und AREA-Anweisungen).

Im einzelnen legen Sie bei der KDCDEF-Generierung für die Lademodule Folgendes fest:

- wann die Lademodule geladen werden sollen

  Für Teilprogramme, d.h. Module, die mit der PROGRAM-Anweisung generiert wurden, können Sie zwischen dem Startzeitpunkt der Anwendung und dem Aufrufzeitpunkt des Teilprogramms wählen.

  Nicht-shareable Teilprogramme können entweder statisch zum Anwendungsprogramm gebunden, als Lademodul beim Anwendungsstart dynamisch nachgeladen oder als Lademodul zum Zeitpunkt des Teilprogrammaufrufs geladen werden.

  Der Private Slice zu einem shareable Teil, der in einem Common Memory Pool steht, kann ebenfalls entweder beim Anwendungsstart oder erst zum Zeitpunkt des Teilprogrammaufrufs geladen werden.

  Der Private Slice zu einem shareable Teil, der sich in nichtprivilegierten Subsystemen befindet, kann man alternativ zum statischen Teil des Anwendungsprogramms binden.

  Module und Datenbereiche, die mit der AREA-Anweisung generiert wurden, müssen entweder statisch zum Anwendungsprogramm gebunden oder zum Zeitpunkt des Starts der Anwendung nachgeladen werden, da der Zugriff auf diese Anwendungsteile nicht unter der Kontrolle von *open*UTM erfolgt.

- aus welchen Bibliotheken die Lademodule geladen werden sollen

  Sie können Bindemodulbibliotheken (OML) oder Programmbibliotheken (PL), die Elemente vom Typ R oder L enthalten, angeben.

- welche Version eines Lademoduls geladen werden soll

  In einer Programmbibliothek können gleichzeitig mehrere Versionen eines Elements enthalten sein. Mit der Versionsnummer legen Sie fest, welche Version eines Elements geladen werden soll.

- Speicherbereich, in den die Lademodule geladen werden sollen

  Ein Lademodul kann entweder in einen Common Memory Pool oder in den Standardkontext der Anwendung geladen werden. Der Standardkontext enthält den statisch gebundenen Teil des Anwendungsprogramms, der mit dem Kommando START-PROGRAM (LOAD-PROGRAM) geladen wurde, sowie die Teile des Anwendungsprogramms, die nicht in Common Memory Pools oder in den Systemspeicher geladen wurden.

  Enthält ein LLM Public und Private Slices, dann wird der Public Slice in einen Common Memory Pool und der Private Slice in den Standardkontext im task-lokalen Speicher geladen.

- ob mit Autolink gebunden werden soll

  Die shareable Teile der Lademodule werden stets ohne die Autolink-Funktion geladen. Bei den nicht-shareable Teilen können Sie über die LOAD-MODULE-Anweisung steuern, ob mit oder ohne Autolink-Funktion geladen werden soll.

  Werden Autolink-Funktion des BLS beim Nachladen und beim Programmaustausch unterdrückt, dann wird der Ladevorgang für Lademodule beschleunigt und Inkonsistenzen in dem geladenen Anwendungsprogramm vermieden. In diesem Fall dürfen die Lademodule ausschließlich offene Externverweise auf Programmteile besitzen, die zum Zeitpunkt des Ladens dieses Moduls bereits im Arbeitsspeicher vorhanden sind.

  Wird mit Autolink geladen, dann werden bei einem Austausch des Lademoduls die Module, die zusätzlich benötigt werden, nachgebunden. Die Autolink-Funktion sollte deshalb nur für Module des Laufzeitsystems, *nicht* jedoch für benutzereigene Module verwendet werden, da per Autolink geladene Module bei einem nachfolgenden Austausch nicht mehr entladen werden.

Desweiteren wird bei der Generierung die Zuordnung von Objekten (PROGRAM-, AREA-Anweisung) zu Lademodulen (LOAD-MODULE-Anweisung) festgelegt. Ob die mit KDCDEF definierte Zuordnung der tatsächlichen Aufteilung der Lademodule in den Bibliotheken entspricht, kann von *open*UTM nicht verifiziert werden. *open*UTM verlässt sich beim Programmaustausch auf die bei der Generierung gemachten Angaben.

Um beim Programmaustausch die BLS-Funktionen nutzen zu können, müssen Sie das Start-LLM mit dem BINDER binden. Auch Lademodule sollten Sie nach Möglichkeit mit dem BINDER binden. Sehen Sie dazu Abschnitt „Binden mit BINDER“ auf Seite 39.

Mit der Reihenfolge, mit der Sie die Lademodule generieren, bestimmen Sie die Reihenfolge, in der die Lademodule geladen werden. Wenn Sie Lademodule, die mit POOL oder STARTUP generiert wurden, ohne die Autolink-Funktion des BLS laden lassen, ist die Reihenfolge der Generierung ausschlaggebend für die Auflösung der unbefriedigten Externverweise beim Laden.

Die Zuordnung von nicht statisch gebundenen Objekten (Teilprogramme und Areas) zu den Lademodulen wird ebenfalls bei der Generierung festgelegt (KDCDEF-Anweisungen PROGRAM und AREA).



Ausführliche Informationen zum Generieren der Lademodule finden Sie im *open*UTM-Handbuch „Anwendungen generieren“.

Hinweise zum Programmaustausch

Mit der Generierung der Lademodule legen sie fest, welche Programmteile später ausgetauscht werden können. Folgendes ist zu beachten:

- Die Funktion „Programmaustausch“ können Sie nur nutzen, wenn Sie das Start-LLM mit dem BINDER binden.
- Programmteile, die später ausgetauscht werden sollen, dürfen nicht statisch ins Anwendungsprogramm eingebunden oder in anwendungsglobale Common Memory Pools oder in den Systemspeicher geladen werden.
- Das BLS unterstützt nur das Entladen von Programmteilen, die mit **einem** Ladevorgang geladen werden. Wenn also eine Gruppe von Modulen ausgetauscht werden soll, müssen Sie diese Module als LLM oder OM vorbinden. Soll ein Einzelmodul ausgetauscht werden, können Sie dieses Modul als LLM oder OM bereitstellen. Es darf dann nicht mit anderen Modulen zusammen gebunden werden.
- Ein Lademodul, das das Administrationsprogramm KDCADM, die Event-Exits START, SHUT, INPUT und FORMAT oder die Event-Services BADTAC, MSGTAC und SIGNON enthält, darf im Betrieb nicht ausgetauscht werden.
- Bei einer Änderungsgenerierung werden auf Anforderung auch die aktuellen Versionsnummern der ausgetauschten Lademodule aus der alten KDCFILE in die neue KDCFILE übertragen.

2.2 Module laden

In welcher Reihenfolge die verschiedenen Programmteile der UTM-Anwendung geladen und wie dabei die Externverweise aufgelöst werden, wird in Bild 1 dargestellt und anschließend beschrieben. Die Nummern bezeichnen dabei jeweils die Reihenfolge, in der die Programmteile geladen werden (müssen). Die Pfeile geben an, in welcher Richtung beim Laden die unbefriedigten Externverweise der Lademodule aufgelöst werden, wenn ohne Autolink geladen wird.

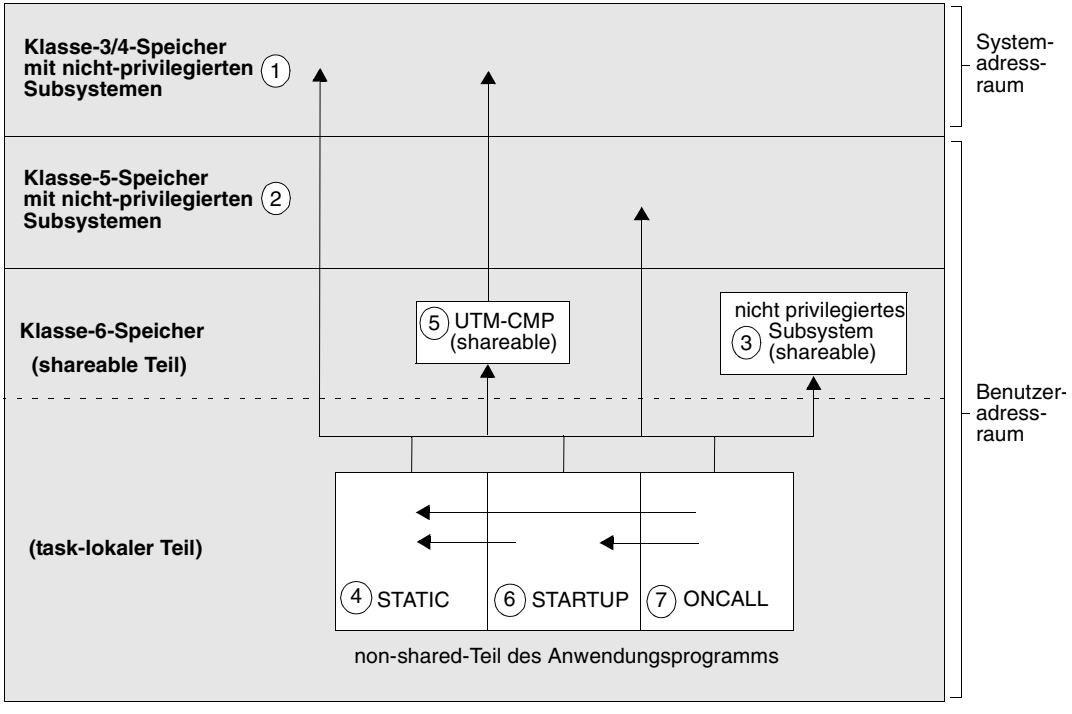


Bild 1: Speicherstruktur einer UTM-Anwendung (CMP = Common Memory Pool)

Mit Ausnahme des Ladens der Subsysteme (durch den Systemadministrator) und des Ladens des statischen Teils (durch ein START-PROGRAM-Kommando) werden alle Ladevorgänge durch *open*UTM angestoßen und kontrolliert:

(1) (2) (3) Gemeinsam benutzbare Module lädt der Systemadministrator vor dem Start der UTM-Anwendung.

④ Das Start-LLM mit allen eingebundenen Lademodulen wird mit dem START-PROGRAM- oder LOAD-PROGRAM-Kommando geladen. Offene Externverweise sollten wegen der Ladeperformance nur aus dem Shared Code befriedigt werden, können aber auch aus Bibliotheken befriedigt werden, wenn Sie im START-PROGRAM- oder LOAD-PROGRAM-Kommando den folgenden Operanden angeben: *MODULE(...,RUN-MODE=ADVANCED(ALTERNATE-LIBRARIES=YES,...))

⑤ Alle Lademodule, die mit LOAD-MODE=POOL und SCOPE=GLOBAL generiert wurden, werden in der Reihenfolge der MPOOL-Anweisungen geladen.
Danach werden alle Lademodule geladen, die mit LOAD-MODULE=POOL und SCOPE=GROUP generiert wurden.

Innerhalb eines Pools werden die Lademodule in der Reihenfolge der LOAD-MODULE-Anweisungen zu diesem Pool geladen.

Beim Laden werden Externverweise nur aus dem Systemspeicher und aus dem eigenen Memory Pool befriedigt; beim Laden von Shared Code wird die Autolink-Funktion unterdrückt.

⑥ Die Lademodule, die mit LOAD-MODE=STARTUP generiert wurden, werden beim Start des Anwendungsprogramms dynamisch nachgeladen. Die Reihenfolge der LOAD-MODULE-Anweisungen bei der Generierung bestimmt auch hier die Reihenfolge beim Laden.

Offene Externverweise können aus dem Systemspeicher, aus den Common Memory Pools und den bis dahin bereits geladenen Modulen befriedigt werden. Externverweise auf die Laufzeitsysteme können durch Nachladen von Modulen der Laufzeitsysteme befriedigt werden (Angabe von ALTERNATE-LIBRARIES=YES in der LOAD-MODULE-Anweisung).
Das gleiche gilt für das Laden der Private Slices von Lademodulen, die mit LOAD-MODE=(POOL,...,START-UP | ONCALL) generiert wurden.

⑦ Lademodule, die mit LOAD-MODE=ONCALL generiert wurden, werden beim erstmaligen Aufruf eines zugeordneten Teilprogramms geladen. Offene Externverweise werden wie die bei STARTUP generierten Lademodule (siehe 6) aufgelöst.

2.3 Empfehlungen für die Strukturierung der Anwendung

Die Ladestruktur einer Anwendung bestimmt maßgeblich die Zeit, die zum Starten der ersten Task und der Folgetasks einer Anwendung benötigt wird. Ebenso legen Sie damit fest, welche Programmteile später während des Anwendungslaufs ausgetauscht werden können. Deshalb sollten Sie folgende Empfehlungen beachten:

- Der mit dem Kommando START-PROGRAM zu ladende statisch gebundene Teil des Anwendungsprogramms (Start-LLM) sollte möglichst klein gehalten werden. Er muss die Systemmodule des KDCROOT und die von diesen Modulen benötigten Laufzeitmodule enthalten.
- Die Administrationsmodule (z.B. das Administrationsprogramm KDCADM) sollten entweder statisch zum Start-LLM oder zu einem eigenen Lademodul gebunden werden. Dieses Lademodul muss beim Start der Anwendung geladen werden, d.h. es muss mit LOAD-MODULE ...,LOAD-MODE=STARTUP generiert werden.
  Das Lademodul, das das Administrationsprogramm KDCADM enthält, darf im Betrieb nicht ausgetauscht werden.
- Die Event-Exits START, SHUT, INPUT und FORMAT und die Event-Services BADTAC, MSGTAC und SIGNON sollten entweder statisch zum Start-LLM oder zu einem eigenen Lademodul gebunden werden. Dieses Lademodul muss beim Start der Anwendung geladen werden, d.h. es muss mit LOAD-MODULE ...,LOAD-MODE=STARTUP generiert werden. Es sollte im Betrieb nicht ausgetauscht werden, da Fehler beim Austausch dieses Moduls zu einem abnormalen Anwendungsende führen können.
- Programmteile, die später ausgetauscht werden sollen, dürfen nicht statisch gebunden oder in anwendungsglobale Common Memory Pools oder in den Systemspeicher geladen werden.
- Programmteile, die von mehr als einer Anwendung benötigt werden, sollten shareable, z.B. als nicht-privilegiertes Subsystem, geladen werden.
- Alle übrigen Anwendungsteile, die shareable sind, sollten Sie zu einem Nachlademodul vorbinden und diesen in einen Common Memory Pool laden. Werden aus bestimmten Gründen mehrere Lademodule mit shareable Programmteilen benötigt, so sollten diese trotzdem nur in **einen** Common Memory Pool geladen werden.

- Viele Compiler erlauben es, das shareable Modul mit dem nicht-shareable Modul gemeinsam in einem LLM abzulegen. Dies vereinfacht die Verwaltung der Module, da sich beide Teile immer in einem Container befinden. In diesem Fall sollten Sie die shareable und die nicht-shareable Teile nicht getrennt vorbinden, sondern das gesamte LLM mit Public und Private Slices, wie es vom Compiler erzeugt wurde, in ein oder mehrere LLMs vorbinden.

  *open*UTM lädt den Public Slice eines solchen LLM in einen Memory Pool und den Private Slice beim Start der Anwendung bzw. beim Aufruf eines Programms aus einem solchen LLM in den task-lokalen Speicher, wenn Sie das LLM mit
  LOAD-MODULE ..., LOAD-MODE=(POOL,...,STARTUP) bzw.
  LOAD-MODE= (POOL,...,ONCALL) generieren.

- Programmteile, die nur gelegentlich benötigt werden, sollten so generiert werden, dass sie erst zum Aufrufzeitpunkt nachgeladen werden
  (LOAD-MODULE ..., LOAD-MODE= ONCALL). Dadurch kann die Startphase verkürzt werden. Diese Lademodule sollten Sie zu kleinen logischen Einheiten vorbinden, da das Laden kleiner Module weniger Zeit benötigt als das Laden großer Module.

- Anwendungsteile, die logisch zusammengehören, sollten Sie zu einem Lademodul vorbinden. Das können z.B. Programmteile sein, die sich gegenseitig referenzieren, oder die zu einer Folge von Teilprogrammläufen in einem Vorgang gehören.

- Alle Programmteile, die nachgeladen werden sollen, sollten Sie als LLM vorbinden. Der Start der Anwendung sowie das Laden von Lademodulen wird dadurch erheblich beschleunigt, da der dynamische Bindelader (DBL) LLMs wesentlich performanter verarbeiten kann als OMs.

- Die Anzahl der Lademodule, die zum Zeitpunkt des Starts des Anwendungsprogramms geladen werden (LOAD-MODE=STARTUP oder LOAD-MODE=(POOL,...,STARTUP), sollte so klein wie möglich gehalten werden. Aus diesem Grund sollten Sie nur die Teile der Anwendung STARTUP generieren, die häufig benötigt werden.
  Weil das Laden weniger großer Lademodule weniger Zeit benötigt als das Laden vieler kleiner Lademodule, sollten diese Teile des Anwendungsprogramms zu größeren Lademodulen vorgebunden werden.

- Es sollte nur **ein** anwendungslokaler Common Memory Pool
  (MPOOL ...,SCOPE=GROUP) definiert werden. In diesen können Sie mehrere vorgebundene Lademodule laden. Damit wird die Zeit zum Einrichten und Laden der Common Memory Pools verkürzt und somit die für den Start der Anwendung benötigte Zeit minimiert.

- OMs und LLMs sollten Sie in unterschiedlichen Programmbibliotheken bereitstellen, da auf diese Weise der Suchalgorithmus des BLS optimiert wird.

- Module, die weder Teilprogramme des Anwendungsprogramms noch Datenbereiche (AREA) sind (z.B. die Module der Laufzeitsysteme der Programmiersprache), brauchen Sie nicht einzeln mit dem Generierungstool KDCDEF als nachzuladende Module zu deklarieren, selbst wenn diese Module nicht statisch eingebunden sind. Sie können diese Module zu größeren Lademodulen (LLM) vorbinden und müssen nur den Namen des Lademoduls in der LOAD-MODULE-Anweisung generieren.



ACHTUNG!
Ob die mit der Anweisung LOAD-MODULE und den Operanden LOAD-MODULE in der PROGRAM- und AREA-Anweisung definierte Zuordnung der tatsächlichen Aufteilung der Lademodule in den Bibliotheken entspricht, kann von *open*UTM nicht verifiziert werden.
*open*UTM verlässt sich beim Nachladen der Lademodule auf die in der Generierung gemachten Angaben. Sie müssen deshalb sicherstellen, dass die von Ihnen verwendeten Bindeprozeduren für die einzelnen Teile des Anwendungsprogramms mit den in der Generierung gemachten Angaben übereinstimmen. Andernfalls kann *open*UTM nicht sicherstellen, dass mit einem bestimmten Lademodul das gewünschte Programm in den Arbeitsspeicher geladen wird.

2.4 Regeln und Einschränkungen beim Einsatz von BLS

Folgende Regeln und Einschränkungen müssen Sie bei der Aufteilung der Objekte in Lademodule sowie beim Binden und Nachladen der Objekte mit dem BLS beachten:

- Die Event-Exits START, SHUT, INPUT und FORMAT und die Event-Services BADTAC, MSGTAC und SIGNON dürfen Sie keinem Lademodul zuordnen, das erst beim Aufruf eines Teilprogramms geladen wird, d.h. mit LOAD-MODE=ONCALL generiert ist. Der Grund dafür ist, dass die Event-Exits und -Services in jedem Prozess der Anwendung verfügbar sein müssen. Sind sie es nicht, dann bricht *open*UTM den Start des Prozesses ab.
  Die Lademodule, die die Event-Exits und -Services enthalten, sollten im Betrieb nicht ausgetauscht werden. Fehler beim Austausch der Lademodule können zu einem abnormalen Anwendungsende führen.
- Das Administrationsprogramm KDCADM dürfen Sie keinem Lademodul zuordnen, das mit LOAD-MODE=ONCALL generiert ist.
  Das Lademodul, in das das Administrationsprogramm KDCADM eingebunden ist, darf nicht ausgetauscht werden.
- Das Start-LLM müssen Sie in einer Programmbibliothek bereitstellen. Es kann nicht in einer Datei stehen.
- Im Kommando START-PROGRAM müssen Sie die folgenden Operanden angeben:

  ```
  RUN-MODE=ADVANCED(UNRESOLVED-EXTRNS=DELAY,
                    [PROGRAM-MODE=ANY,]
                    LOAD-INFORMATION=REFERENCES)
  ```

  Die Angabe von PROGRAM-MODE=ANY ist erforderlich, wenn die Anwendung in den oberen Adressraum geladen werden soll.
- Wenn Sie per Administration einen Programmaustausch anfordern, müssen Sie explizit die Version des neu zu ladenden Moduls angeben.
- Für Lademodule dürfen Sie nur die Namen von OMs oder LLMs angeben. Aus Performancegründen unterstützt *open*UTM nicht das Nachladen über CSECT- oder ENTRY-Namen.
- BLS unterstützt pro BS2000-Benutzerkennung maximal acht Common Memory Pools mit SCOPE=GROUP und acht Common Memory Pools mit SCOPE=GLOBAL.
- Zwei UTM-Anwendungen sollten möglichst unter unterschiedlichen BS2000-Benutzerkennungen gestartet werden, um Fehler zu vermeiden, die wegen gleicher Modulnamen in shareable Teilen auftreten können. Module, die von mehreren Anwendungen benutzt werden, sollten daher in anwendungsglobale Common Memory Pools oder in nichtprivilegierte Subsysteme geladen werden.

- Durch einen Programmaustausch kann kein neu erzeugter Tabellenmodul aktiviert werden.
- Die Variantennummer des LMS ist beim Laden von Lademodulen mit BLS ohne Bedeutung.
- BLS unterstützt nur das Entladen von Programmteilen, die in **einem** Ladevorgang geladen wurden. Deshalb müssen Programmteile, die als separate Teile austauschbar sein sollen, in der Form bereitgestellt werden, in der sie geladen und ausgetauscht werden sollen. Soll ein Einzelmodul ausgetauscht werden, dann kann dieses Modul als OM oder LLM bereitgestellt werden; soll eine Gruppe von Lademodulen ausgetauscht werden, dann sind diese Lademodule als LLM vorzubinden.
- Lademodule, die TCB-Entries enthalten, dürfen im laufenden Betrieb nicht ausgetauscht werden.
- Module, die über die Autolink-Funktion nachgeladen wurden, können nicht ausgetauscht oder entladen werden. Ausnahme: Beim Austausch der gesamten Anwendung (z.B. mit KDCAPPL PROG=NEW) werden *alle* Anwendungsteile neu geladen.
- FHS V8.2 wurde auf BLS umgestellt und kann daher nicht mehr aus der TASKLIB nachgeladen werden. FHS-Format-Module können ab dieser Version ausgetauscht werden.
- Beim Vorbinden der Lademodule zu LLMs ist zu beachten, dass die eingebundenen Elemente nicht in mehreren Lademodulen vorkommen sollten. Aus diesem Grund sind insbesondere beim Einbinden der Laufzeitsysteme der Programmiersprachen Einschränkungen zu beachten. Wie Sie die für das Anwendungsprogramm benötigten Module der Laufzeitsysteme einbinden können, ist im Abschnitt „Laufzeitsysteme binden“ auf Seite 46 beschrieben.

2.5 Shared Code nutzen

Viele Compiler bieten eine Option an, die es gestattet, bei der Übersetzung von Programmen einen shareable Teil zu erzeugen. Dieser muss nicht unbedingt in einem eigenen Objektmodul abgelegt werden, sondern kann zusammen mit dem nicht-shareable Teil in einem LLM stehen, der in einen Public und einen Private Slice unterteilt ist.

Folgende Objekte können shareable geladen werden:

- shareable Module der Teilprogramme
- Formate
- ein Teil der Module für die Formatierung (falls shareable)
- gemeinsam verwendbare Datenbereiche (Areas)
- der Datenbank-Verbindungsmodul, falls dieser shareable ist
- der Meldungsmodul
- Module der Laufzeitsysteme der Programmiersprachen, die shareable sind (siehe Handbücher der einzelnen Programmiersprachen)

Wenn Teile eines Teilprogramms shareable sein sollen, so müssen Sie das bereits beim Programmieren berücksichtigen. Was Sie beim Programmieren beachten müssen, ist abhängig vom Compiler, der das Teilprogramm übersetzt. Näheres dazu finden Sie im *open*UTM-Handbuch „Anwendungen programmieren mit KDCS" bzw. in der entsprechenden Sprachergänzung.

COBOL-Teilprogramme, die mit dem COBOL85-Compiler übersetzt werden, sind shareable, wenn Sie bei der Übersetzung die Option `COMOPT GENSHARE=YES` angeben. Mit der Option `COMOPT GEN-LLM=YES` veranlassen Sie den Compiler, die Objekte in einen LLM mit Slices abzulegen.

C/C++-Teilprogramme sind shareable, wenn Sie beim Übersetzen die folgende Option angeben:

```
COMPILER-ACTION=MODULE-GENERATION(SHAREABLE-CODE=YES,...)
```

Die entsprechende Option in weiteren Programmiersprachen finden Sie in den zugehörigen Benutzerhandbüchern.

Außerdem werden auch viele Laufzeitsysteme der Programmiersprachen mit einem shareable Teil ausgeliefert.

i Wenn Sie shareable Teile z.B. eines Teilprogramms austauschen wollen, das nicht-shareable Teile enthält, dann sollten Sie die shareable und nicht-shareable Teile zu einem LLM mit Slices vorbinden. Andernfalls können beim Austausch Konsistenzlücken entstehen. Konsistenzprobleme können Sie auch dadurch umgehen, indem Sie vor dem Austausch eines Lademoduls die TACs der in diesem Lademodul enthaltenen Teilprogramme sperren und diese erst nach dem Austausch der shareable und der nicht-shareable Teile wieder freigeben.

Shared Code laden

Die shareable Programmteile brauchen für alle Tasks der Anwendung(en) gemeinsam nur einmal geladen werden. In den task-lokalen Speicher müssen dann nur noch die non-shareable Teile geladen werden.

*open*UTM bietet verschiedene Möglichkeiten, shareable Objekte zu laden:

- in den **Systemspeicher** als nicht-privilegiertes Subsystem in den Klasse-3/4/5- oder 6-Speicher,
- in einen von *open*UTM verwalteten **Common Memory Pool** im Benutzerspeicher (Klasse-6-Speicher).

Sehen Sie dazu auch das Handbuch „BS2000/OSD Verwaltung von Subsystemen (DSSM/SSCM)“.

2.5.1 Shared Code im Systemspeicher

Shareable Teile der Anwendungsteilprogramme und Teile der Laufzeitsysteme können mit Hilfe der im BS2000/OSD vorgesehenen Schnittstellen als shareable Programme in nicht-privilegierte Subsysteme geladen werden.

Shared Code, der in nicht-privilegierte Subsysteme geladen ist, kann während des Betriebs der Anwendung ausgetauscht werden.

Die **shareable** Module müssen durch den Administrator in den Speicher geladen werden, bevor die Anwendung gestartet wird. Sehen Sie hierzu auch die Hinweise am Ende dieses Abschnitts. Im Gegensatz zu Shared Code in Common Memory Pools, der unter Kontrolle von *open*UTM geladen und ausgetauscht werden kann, können Sie die shareable Module im Systemspeicher nicht selbst verwalten.

Non-shareable Teile der Teilprogramme müssen mit der PROGRAM-Anweisung beschrieben und ggf. mit der LOAD-MODULE-Anweisung einem Lademodul zugeordnet werden. Das Lademodul bzw. dessen private Slice wird beim Start des Anwendungsprogramms dynamisch in den task-lokalen Speicher (Klasse-6-Speicher) geladen. Über die Externbezüge zu den shareable Modulen werden die Verknüpfungen in den Shared Code dynamisch hergestellt.
Die Lademodule (OM-Format), die die shareable Module des Teilprogramms enthalten, müssen nicht zusammen mit den Lademodulen, die die nicht-shareable Programmteile enthalten, in einer Programmbibliothek stehen.

Hinweise

- Subsysteme sollten im LLM-Format erstellt werden, da diese Subsysteme im laufenden Betrieb der Anwendung **ohne** Konsistenzlücke ausgetauscht werden können. Subsysteme im LLM-Format können nur für BLS-Anwendungen erstellt werden.
  Beim Austausch des Subsystems wird eventuell der Public Slice des alten LLMs in den tasklokalen Teil des Anwendungsprogramms geladen, bis auch der Private Slice ausgetauscht wird (z.B. mit KDCPROG initiiert).

  Subsysteme im OM-Format haben hingegen beim Austausch im laufenden Betrieb eine Konsistenzlücke, die zwischen dem DSSM-Subsystemkommando

  ```
  START-SUBSYSTEM SUBSYSTEM-NAME=subsystemname          -
                , VERSION=new-version                   -
                . VERSION-PARALLELISM=*EXCHANGE-MODE
  ```

  und dem UTM-Administrationskommando (oder einen entsprechenden KDCADMI-Aufruf an der Programmschnittstelle zur Administration)

  ```
  KDCPROG LOAD-MODULE=load-module ,VERSION=new-version
  ```

  existiert. Hier kann es beispielsweise vorkommen, dass nach dem DSSM-Kommando und vor dem UTM-Administrationskommando ein UTM-Anwendungsprogramm mit PEND ER beendet wird und die betroffene Task des UTM-Anwendungsprogramms mit dem alten Private Slice und einem neuen Public Slice in einem neuen Subsystem startet.

- Beim Austausch von Subsystemen verschiedener Versionen, deren Module in der gleichen Bibliothek stehen (Bedingung zum Austausch von Lademodulen bei *open*UTM), ist darauf zu achten, dass unterschiedliche LINK-ENTRYs angegeben werden, da sonst das DSSM ggf. den Austausch nicht durchführen kann.
- Die Beispielprozedur MAKE-SUBSYSTEM inklusive Erläuterung finden Sie in der Bibliothek SYSLIB.UTM.052.EXAMPLE.
- Beim Erstellen eines Subsystems sollte das LLM-Format 2 verwendet werden, da hier nur ein Subsystemeintrag angegeben werden muss.

2.5.2 Shared Code in Common Memory Pools

Ein Common Memory Pool ist ein Bereich im Benutzerspeicher (Klasse-6-Speicher). In einen Common Memory Pool können Sie Objekte laden, die beim Binden des Anwendungsprogramms nicht statisch dazugebunden wurden. *open*UTM bietet zwei Möglichkeiten, einen Common Memory Pool einzurichten:

- als anwendungslokalen Pool

  Auf diesen Pool können alle Teilprogramme einer Anwendung zugreifen. Es können auch Teilprogramme anderer Anwendungen auf den Pool zugreifen, sofern diese Anwendungen unter derselben BS2000-Benutzerkennung gestartet wurden.

- als anwendungsglobalen Pool

  Teilprogramme **mehrerer** Anwendungen können auf den gleichen Common Memory Pool zugreifen, unabhängig von den Benutzerkennungen, unter denen diese Anwendungen gestartet wurden.

Jeden Common Memory Pool müssen Sie mit der KDCDEF-Anweisung MPOOL generieren. In der MPOOL-Anweisung legen Sie fest, ob ein Common Memory Pool anwendungsglobal oder anwendungslokal eingerichtet wird.

2.5.2.1 Anwendungslokaler Pool

Ein anwendungslokaler Pool wird von der ersten Task einer Anwendung angelegt. Alle weiteren Tasks dieser Anwendung greifen ebenfalls darauf zu, d.h. der Pool wird im Sinne der Memory-Pool-Verwaltung des BS2000/OSD mit SCOPE=GROUP angelegt. Auch Tasks weiterer UTM-Anwendungen, die unter derselben Benutzerkennung gestartet wurden, können sich an diesen Common Memory Pool anschließen.

Der anwendungslokale Pool bietet folgende Vorteile für den Anwender:

- Die Nutzung der in diesen Pool geladenen Programme ist nur von dieser Benutzerkennung aus möglich.
- Der Benutzer hat die volle Kontrolle über die Verwendung von shareable Objekten, unabhängig von der System-Initialisierung.
- Der Pool existiert nur so lange die Anwendung läuft, bzw. wenn mehrere Anwendungen auf den Pool zugreifen, bis die letzte Anwendung beendet wird. Mit dem Ende der Anwendung wird der Pool aufgelöst und die Programme werden - wie alle anderen Anwenderprogramme auch - entladen.
- Die in einen Common Memory Pool geladenen Programme können unter bestimmten Voraussetzungen während eines Anwendungslaufs dynamisch ausgetauscht werden. Näheres hierzu finden Sie in Kapitel „Programmaustausch im Betrieb“ auf Seite 147.
- Der physikalische Speicher wird besser ausgenutzt.

Mit dem Beenden der Anwendung wird der Pool aufgelöst und die Programme werden - wie alle anderen Anwenderprogramme auch - entladen.



ACHTUNG!
Alle Module, Formate und Datenbereiche haben innerhalb eines Pools die gleiche Zugriffsberechtigung.

Programme in Common Memory Pools dürfen keine offenen Externverweise auf Adressen außerhalb des Common Memory Pools enthalten, die auf Module in anderen Common Memory Pools oder nicht-privilegierten Subsystemen im Klasse 5/6-Speicher zeigen. Bei Nutzung der BLS-Schnittstelle werden diese CSECTs und ENTRYs nicht gefunden. Weiterhin dürfen offene Externverweise nur auf Shared Code zeigen, der für alle Tasks auf derselben Adresse liegt. Sehen Sie dazu die DSSM-Handbücher.

2.5.2.2 Anwendungsglobaler Pool

Ein anwendungsglobaler Pool wird für Module verwendet, die in mehreren Anwendungen gebraucht werden wie z.B. Formate, Verbindungsmodul für Formatierung, Datenbank-Verbindungsmodul, Sprach-Laufzeitmodule, gemeinsam nutzbare Datenbereiche, aber auch Routinen in Teilprogrammen und der Meldungsmodul.

Der anwendungsglobale Pool wird von der ersten Task der ersten UTM-Anwendung eingerichtet. An diesen Pool können sich alle weiteren Tasks derselben und anderer Anwendungen anschließen, d.h. der Pool wird im Sinne der Memory-Pool-Verwaltung des BS2000/OSD mit SCOPE=GLOBAL angelegt.

Ein anwendungsglobaler Pool bietet ähnliche Vorteile wie ein anwendungslokaler Pool. Er hat jedoch den Nachteil, dass die in einen solchen Pool geladenen Programme während eines Anwendungslaufs nicht dynamisch ausgetauscht werden können.

Werden globale Common Memory Pools gleichen Inhalts/Namens in mehreren UTM-Anwendungen verwendet, muss in der KDCDEF-Anweisung MPOOL der Parameter PAGE=X'xxxxxxxx' mit gleicher Adresse in allen Anwendungen angegeben werden. Die mit PAGE= angegebene Adresse ist so zu wählen, dass der damit reservierte Adressbereich in all diesen Anwendungen verfügbar ist.

Beispiel

Anwendung 1 verwendet den globalen Pool MPEINS. Anwendung 2 verwendet ebenfalls diesen Pool und zusätzlich den Pool MPZWEI. Folgende KDCDEF-Steueranweisungen werden benötigt:

Generierung Anwendung 1:

```
MPOOL MPEINS, SCOPE=CLOBAL, PAGE=X'01000000', SIZE=...
```

Generierung Anwendung 2:

```
MPOOL MPZWEI, SCOPE=CLOBAL, SIZE=...
MPOOL MPEINS, SCOPE=CLOBAL, PAGE=X'01000000', SIZE=...
```

Eine Alternative für die Verwendung von PAGE= ist, die gemeinsamen Pools in allen Anwendungen in der gleichen Reihenfolge zu generieren. Die MPOOL-Anweisungen müssen außerdem an erster Stelle stehen.

2.5.2.3 Shareable Objekte generieren, die in einen Common Memory Pool geladen werden

Sollen Teile Ihres Anwendungsprogramms in einen Common Memory Pool geladen werden, dann beachten Sie bitte Folgendes:

- Aus Performancegründen sollten Sie, soweit möglich, alle shareable Teile eines Anwendungsprogramms, die in einen Common Memory Pool geladen werden sollen, zu **einem** Lademodul zusammenfassen.
- Das vom Compiler erzeugte shareable Code-Modul des Programms muss in einem LLM oder OM enthalten sein.
- Erzeugt ein Compiler für den shareable und den nicht-shareable Teil zwei getrennte Objektmodule, dann sollten Sie diese Module mit dem Binder zu einem LLM mit Slices vorbinden.
- LLMs mit Slices können Sie mit einer einzigen LOAD-MODULE-Anweisung generieren:

```
LOAD-MODULE llm-name   ,VERSION=version                                  -
                       ,LOAD-MODE=(POOL,poolname,{STARTUP|ONCALL})       -
                       ,LIB=program-lib                                  -
                       .ALTERNATE-LIBRARIES={YES|NO}
```

  Durch diese Anweisung wird der Public Slice des LLM in den Common Memory Pool *poolname* geladen, der Private Slice wird entweder bei Anwendungsstart (STARTUP) oder bei Programmaufruf (ONCALL) nachgeladen. Für die durch *open*UTM aufzurufenden Programme dieses LLMs sind zusätzlich PROGRAM-Anweisungen notwendig.

  Alternativ dazu können Sie auch das shareable und das nicht-shareable Modul durch zwei LOAD-MODULE-Anweisungen generieren. Dies sollten Sie jedoch möglichst vermeiden, da sich diese beiden Module nicht ohne Konsistenzlücke austauschen lassen.

- Einen gemeinsam nutzbaren Datenbereich (Area), der in den Common Memory Pool geladen werden soll, müssen Sie mit einer AREA-Anweisung beschreiben. Die Area muss dann in dem Lademodul enthalten sein, der folgendermaßen generiert wird:

```
LOAD-MODULE ar-share      ,VERSION=version                                  -
                          ,LOAD-MODE=(POOL,poolname,NO-PRIVATE-SLICE)       -
                          ,LIB=libname
```

  AREAs, denen beim Übersetzen oder durch den Binder das Attribut PUBLIC zugeordnet wurde, können auch zusammen mit anderen Modulen in einen LLM mit Slices vorgebunden werden. Dieses LLM kann dann folgendermaßen generiert werden:

```
LOAD-MODULE llm-with-slices ,VERSION=version                                -
                            ,LOAD-MODE=(POOL,poolname,{STARTUP|ONCALL})     -
                            ,LIB=libname
```

Beispiel

Das Beispiel geht davon aus, dass zum Übersetzen der COBOL85-Compiler verwendet wurde und dass der Compiler die Objekte in ein LLM abgelegt hat. In den anwendungslokalen Pool LCPOOL sollen die shareable Module der COBOL-Teilprogramme TP1 und TP2, die Formatbeschreibungen FORMAT1 und FORMAT2 und der Datenmodul DATAMOD geladen werden. LCPOOL soll auf Adresse X'020000' geladen werden, 128 KB belegen können und schreibgeschützt sein.

```
/MPOOL        LCPOOL,SIZE=2,SCOPE=GROUP,ACCESS=READ,PAGE=X'20000'
/LOAD-MODULE  LLM-LCPOOL,VERSION=1, -
/             LOAD-MODE=(POOL,LCPOOL,STARTUP), -
/             LIB=libname
/PROGRAM      TP1 ,LOAD-MODULE=LLM-LCPOOL,COMP=ILCS
/PROGRAM      TP2 ,LOAD-MODULE=LLM-LCPOOL,COMP=ILCS
/AREA         DATAMOD,LOAD-MODULE=LLM-LCPOOL
```

Die Objektmodule müssen Sie vor dem Start der Anwendung zu dem LLM LLM-LCPOOL vorbinden und dabei in der BINDER-Anweisung START-LLM-CREATION die Option BY-ATTRIBUTES(PUBLIC=YES) angeben, wodurch das LLM in eine Public und einen Private Slice unterteilt wird. Das so erzeugte LLM müssen Sie in der Bibliothek *libname* bereitstellen.

3 Anwendungsprogramm erzeugen

Die Teilprogramme definieren die Anwendungslogik und müssen vor dem Start der Anwendung geschrieben und übersetzt werden. Sehen Sie dazu das *open*UTM-Handbuch „Anwendungen programmieren mit KDCS”.

Damit die Teilprogramme unter *open*UTM ablaufen können, wird das UTM-Anwendungsprogramm wie folgt erzeugt:

- ROOT-Tabellen-Source, die Sie mit dem Generierungstool KDCDEF erzeugt haben, assemblieren.
- ROOT-Systemmodule und die sprachspezifischen Laufzeitsysteme zu einem Start-LLM binden. Es können weitere anwendungsspezifische Teile wie Meldungsmodule, Format-Bibliotheken und Teilprogramme in das Start-LLM eingebunden werden.

Das ROOT-Tabellenmodul kann bei Nutzung des BLS beim Start der Anwendung dynamisch nachgeladen werden.

Die einzelnen Schritte, welche zum Erzeugen eines UTM-Anwendungsprogramms notwendig sind, zeigt die folgende Abbildung.

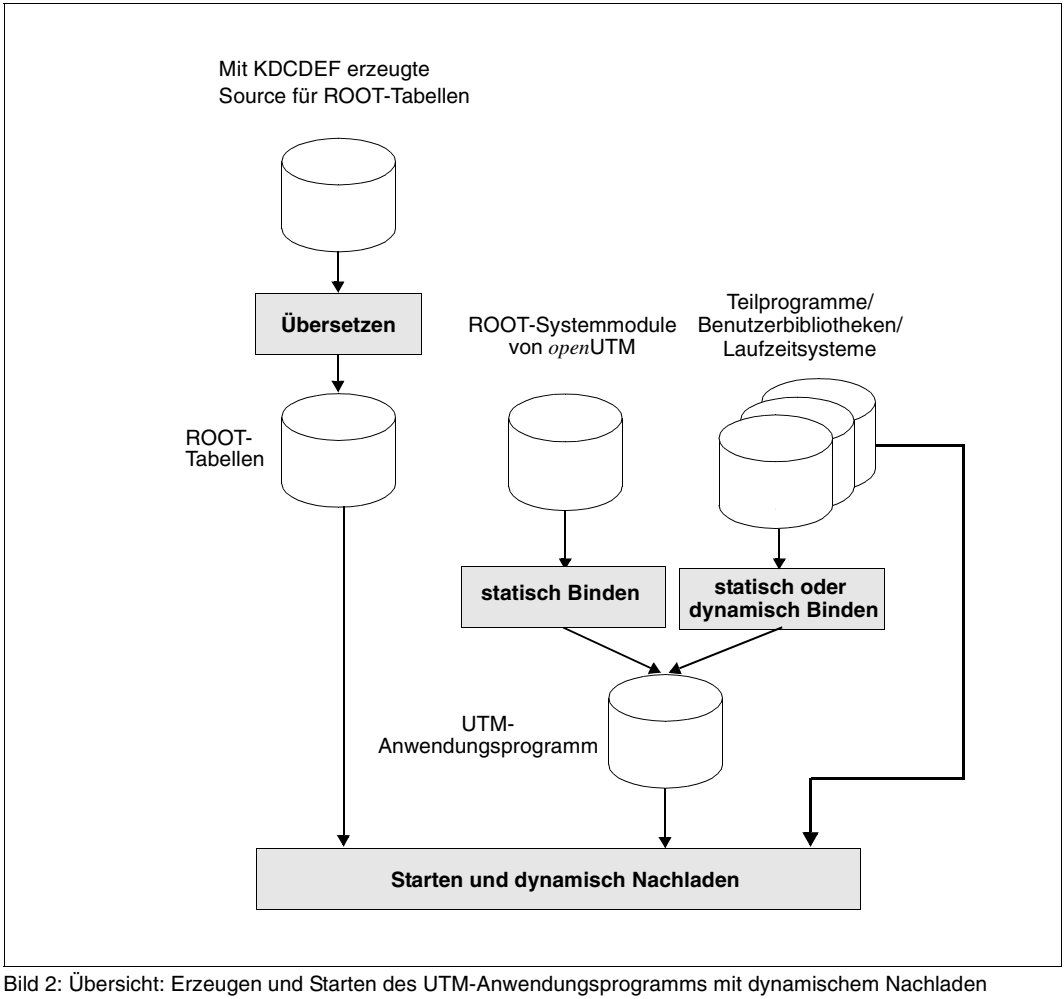


Bild 2: Übersicht: Erzeugen und Starten des UTM-Anwendungsprogramms mit dynamischem Nachladen

Die ROOT-Tabellen können auch statisch zum Anwendungsprogramm gebunden werden. Sehen Sie dazu die folgende Abbildung.

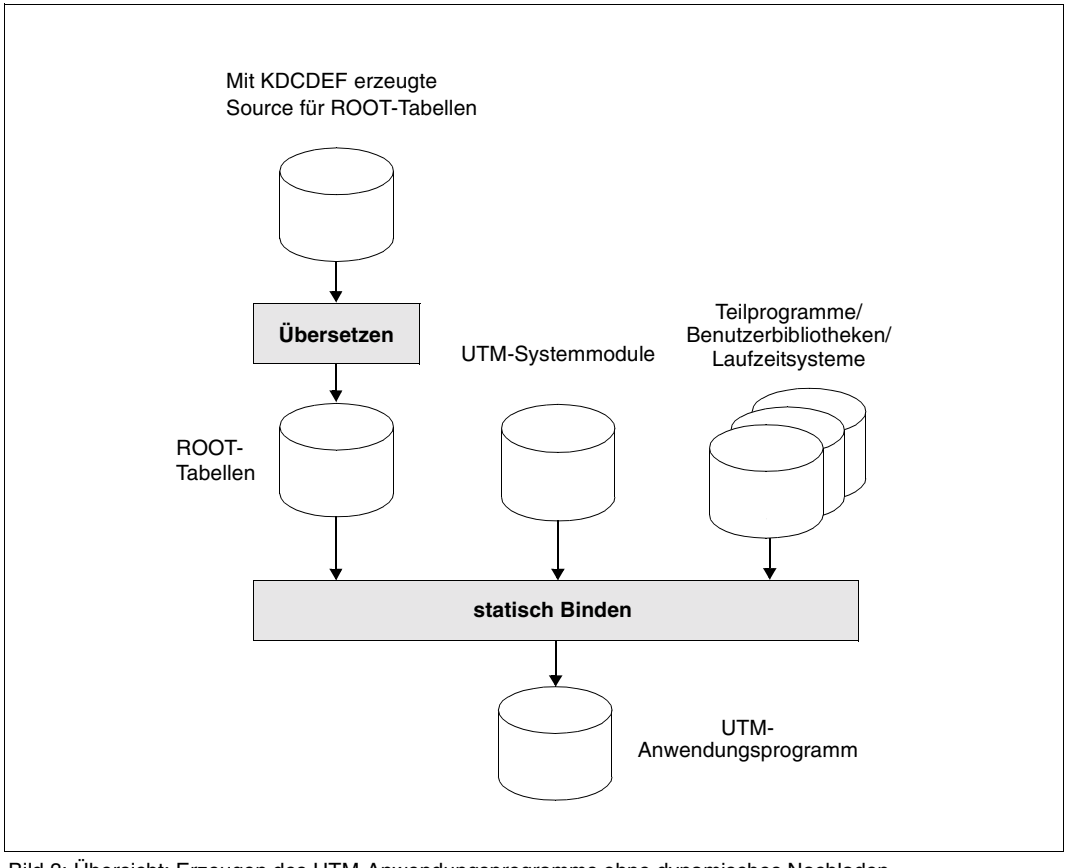


Bild 3: Übersicht: Erzeugen des UTM-Anwendungsprogramms ohne dynamisches Nachladen

Main Routine KDCROOT

Aus den ROOT-Systemmodulen, die mit *open*UTM ausgeliefert werden, entsteht beim Binden die Main Routine KDCROOT als Teil des Anwendungsprogramms. KDCROOT fungiert beim Ablauf der Anwendung als steuerndes Hauptprogramm und übernimmt u.a. folgende Aufgaben:

– Verbindung zwischen den Teilprogrammen und den UTM-Systemfunktionen
– Ablauf-Koordination von Teilprogrammen unterschiedlicher Programmiersprachen
– Kopplung zu Datenbanken
– Zusammenarbeit mit Formatierungssystemen

Weiter enthält KDCROOT Bereiche für variable Daten sowie die Nachrichtenbereiche. Nähere Informationen zur Main Routine KDCROOT sind dem *open*UTM-Handbuch „Anwendungen programmieren mit KDCS” zu entnehmen.

Der Standardname für die ROOT-Tabellen-Source ist ROOT.SRC.ASSEMB.*rootname*. Die von KDCDEF erzeugten Sourcen für die ROOT-Tabellen müssen Sie assemblieren, beispielsweise mit ASSGEN. Die ROOT-Systemmodule, die zum Anwendungsprogramm dazugebunden werden, sind in der Modulbibliothek SYSLNK.UTM.052 enthalten.

3.1 Bestandteile des Anwendungsprogramms

Ein UTM-Anwendungsprogramm besteht aus einer Reihe von Modulen, die spätestens zur Ablaufzeit als ein Programm gebunden sein müssen.

Die folgenden Module sind zum Ablauf notwendig:

- rootmodul

  Übersetztes KDCROOT-Tabellenmodul, das Sie in einer Programmbibliothek oder Objektmodulbibliothek ablegen müssen. Wenn Sie die ROOT-Nachladetechnik verwenden, geben Sie diese Bibliothek beim Startparameter TABLIB= und den Namen des Moduls beim Startparameter ROOTNAME= an.

- Systemmodule des KDCROOT

  Module, die *open*UTM zum Ablauf benötigt. Dazu gehören u.a. die Module KDCRTST, KDCRTCF und KDCRTCU. Sie stehen in der Bibliothek SYSLNK.UTM.052.

- Teilprogramme

  Vom Anwender erstellte Teilprogramme der Anwendung, die Sie übersetzen und in eine oder mehrere Programmbibliotheken oder Objektmodulbibliotheken eintragen lassen müssen.

- Datenbank-Verbindungsmodul

  Dient zum Anschluss von *open*UTM an das Datenbanksystem, das bei der KDCDEF-Generierung im Operanden TYPE der DATABASE-Anweisung angegeben wurde. Wie das DB-System dieses Modul bereitstellt, ist in den Handbüchern der entsprechenden DB-Systeme beschrieben.

- Verbindungsmodul für das Formatierungssystem

  Dient zum Anschluss von *open*UTM an ein Formatierungssystem, das im Operanden TYPE der FORMSYS-Anweisung generiert wurde. Wie das Formatierungssystem dieses Modul bereitstellt, ist in den Handbüchern der entsprechenden Formatierungssysteme beschrieben.

- Administrations- und UTM-SAT-Administrationsteilprogramm

  Programme zur Administration der UTM-Anwendung. Sie können ein solches Programm unter Verwendung der Programmschnittstelle zur Administration entweder selbst schreiben oder die mit *open*UTM ausgelieferten Standard-Administrationsprogramme KDCADM und KDCSADM verwenden. Jedes Administrationsprogramm müssen Sie in einer eigenen PROGRAM-Anweisung generieren. Die Objektmodule der Administrationsprogramme stehen in der Bibliothek SYSLNK.UTM.052 zur Verfügung.

- Laufzeitsystem für den KDCROOT

  wird immer benötigt und wird in der Bibliothek SYSLNK.UTM.052.SPLRTS geliefert. Zusätzlich braucht der KDCROOT eine der Bibliotheken SYSLNK.CRTE bzw. SYSLNK.CRTE.PARTIAL-BIND, wobei die zweite Bibliothek Performancevorteile bringt und daher vorzuziehen ist. CRTE ist Software-Voraussetzung für *open*UTM und enthält das C-Laufzeitsystem, das ILCS und das COBOL85-Laufzeitsystem. Das C-Laufzeitsystem und das ILCS werden von *open*UTM immer benötigt.

- Laufzeitsysteme der Programmiersprachen

  werden benötigt, wenn mindestens ein Teilprogramm der Anwendung in einer höheren Programmiersprache geschrieben wurde. Falls die Anwendung ILCS-fähige Teilprogramme enthält, müssen Sie darauf achten, dass die höchste zur Verfügung stehende ILCS-Version eingebunden wird. Weitere Bindehinweise dazu erhalten Sie in den Informationen zu ILCS im Benutzerhandbuch CRTE. Die ILCS-Module sind in der CRTE-Bibliothek enthalten.

  Welche Laufzeitsysteme mit welchen Compilerversionen verwendet werden dürfen, ist in Abschnitt „Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen“ auf Seite 245 beschrieben.

- Sprachanschlussmodule

  werden für jede generierte Programmiersprache oder deren Linkage benötigt. Für SPL, Assembler, COB1 und ILCS-Linkage sind diese Anschlussmodule schon in *open*UTM eingebunden. Für alle anderen Programmiersprachen müssen Sie die Anschlussmodule spätestens zum Anwendungsstart aus sprachspezifischen Bibliotheken einbinden.

- gemeinsam verwendbare Bereiche

  (siehe AREA-Anweisung von KDCDEF)

- benutzereigene Meldungsmodule

  falls benutzereigene Meldungsmodule statisch eingebunden werden sollen.

3.2 Binden mit BINDER

UTM-Anwendungen können die Funktionalität des BLS zum Laden von LLM (Link and Load Module, Bindelademodule) und von OM (Object Module, Bindemodul) nutzen. Damit bieten sich erweiterte Möglichkeiten:

- bei der Verwaltung der Objekte durch die Nutzung von LLM, die in eine Private und eine Public Slice unterteilt sind.

  Dadurch können shareable und nicht-shareable Teile eines Moduls, die in einem Compilerlauf erzeugt wurden, in einem Container verwaltet werden. Dies vermeidet weitgehend Inkonsistenzen beim Vorbinden von Modulen.

- beim Laden eines Objekts (Programm, Modul, Area).

  Ein Objekt kann statisch eingebunden oder beim Anwendungsstart nachgeladen werden. Für Teilprogramme besteht darüberhinaus die Möglichkeit, sie erst zum Zeitpunkt des ersten Aufrufs nachzuladen.

- beim Programmaustausch.

  Es kann ein einzelnes Objekt (LLM oder OM) oder eine vorgebundene Gruppe von Objekten (LLM oder OM) ausgetauscht werden.

Beim Einsatz des BLS braucht nur ein Teil der Objekte der Anwendung statisch zu dem Anwendungsprogramm gebunden werden. Dieser Teil des Anwendungsprogramms heißt **Start-LLM** und wird mit dem Kommando START-PROGRAM FROM-FILE=*MODULE(...) geladen und gestartet. Zu diesem Teil des Anwendungsprogramms können Sie die Systemmodule des KDCROOT und die Objekte (Teilprogramme, Module und Datenbereiche) einbinden, die keinem Nachlademodul zugeordnet sind.

Das vom KDCDEF erzeugte ROOT-Tabellenmodul muss nicht mehr statisch gebunden werden, d.h. es kann beim Start der Anwendung dynamisch nachgeladen werden.

Die anderen Teile des Anwendungsprogramms können in Form von Lademodulen bereitgestellt werden. Ein Lademodul ist

- entweder ein vorgebundenes Bindelademodul, genannt Link Load Module (=**LLM**), das als Element vom Typ L in einer Programmbibliothek steht,
- oder ein Objektmodul (=**OM**), das als Element vom Typ R in einer Objektmodulbibliothek steht.

Aus Gründen der Performance sollten Sie jedoch alle Lademodule mit dem BINDER zu LLM vorbinden und als Elemente vom Typ L in einer Programmbibliothek bereitstellen.

Bei der Generierung müssen Sie für jedes Bindelademodul und jedes Objektmodul genau eine LOAD-MODULE-Anweisung eingeben und dabei angeben, wann das Modul dazugebunden und wohin es geladen werden soll. Siehe dazu *open*UTM-Handbuch „Anwendungen generieren“. Hinweise zur Umstellung von TSOSLNK / alte DLL-Schnittstelle auf BLS finden Sie auf Seite 55.

Existieren beim Binden der Anwendung mit dem BINDER unbefriedigte Externverweise auf Teilprogramme, lässt sich die Anwendung trotzdem starten. Können diese Externverweise nicht durch dynamisches Nachladen aufgelöst werden, sind die zugeordneten Transaktionscodes für die Anwendung ungültig, d.h. sie können nicht benutzt werden.

3.2.1 LLMs mit Slices

Der BINDER bietet Ihnen die Möglichkeit, CSECTs, die in einem LLM enthalten sind, nach bestimmten Kriterien zu einer ladbaren Einheit zusammenzufassen, die **Slice** genannt wird. Dabei wird zwischen Slices unterschieden, die nach Attributen von CSECTs gebildet werden (Slices by Attributes) und solchen, die vom Benutzer definiert werden (User defined Slices).

*open*UTM kann nur LLMs mit Slices by Attributes verarbeiten und unterstützt dabei das Attribut PUBLIC. Viele Compiler bieten die Funktion an, die erzeugten Objekte in einem LLM mit Public und Private Slice abzulegen. Die Public Slice enthält dann den shareable Teil des Objekts und die Private Slice den Teil, der nicht-shareable ist. Dieses Verfahren vereinfacht die Verwaltung von shareable und nicht-shareable Teilen eines Programms, da sich beide in einem LLM befinden, siehe auch Abschnitt „LLMs mit Public/Private Slice binden“ auf Seite 44. Wenn Sie ein solches LLM bei *open*UTM entsprechend generieren, dann lädt *open*UTM die Public Slice in einen Common Memory Pool und die Private Slice in den task-lokalen Speicher.

Kann beim Laden oder beim Austausch die Public Slice eines LLM nicht in den Common Memory Pool geladen werden, lädt *open*UTM das ganze LLM in den tasklokalen Speicher.

3.2.2 Binden von LLMs

Als LLMs sollten Sie alle Modulgruppen vorbinden, die in einem logischen Zusammenhang stehen. Dies können z.B. sein:

- Teile der Laufzeitsysteme
- logisch zusammengehörige Teile einer UTM-Anwendung, wie z.B. alle Teilprogramme eines oder mehrerer Vorgänge einschließlich Vorgangsexits
- FHS-Formate ab FHS V8.2 samt der zugehörigen Teilprogramme

Damit beim dynamischen Binden eines LLM die Externverweise dieses LLM befriedigt werden können, müssen Sie beim Erzeugen des LLM unbedingt folgende BINDER-Anweisungen angeben:

```
/START-LLM-CREATION    INTERNAL-NAME=int-name           - 1.
/                     ,INTERNAL-VERSION=int-vers        -
/                     ,...

/SET-EXTERN-RESOLUTION SYMBOL-TYPE=*REFERENCES          - 2.
/                     ,RESOLUTION=*STD

/SAVE-LLM          ...,ELEMENT=*INTERNAL-NAME(          - 3.
/                       VERSION=*INTERNAL-VERSION)      -
/                     ,FOR-BS2000-VERSIONS=*FROM-OSD-V4
```

1. Hier legen Sie schon Name und Version des LLM fest, unter denen das LLM zum Abschluss des Bindens in der Bibliothek abgespeichert wird.
2. Die Behandlung unbefriedigter Externverweise wird festgelegt. Die angegebenen Operandenwerte sind die Standardwerte, die auf keinen Fall geändert werden dürfen.
3. Das LLM muss unter dem internen Namen abgespeichert werden; das Format des LLM muss korrekt sein (LLM-Format 2).

   Wird beim Abspeichern ein anderer Name als der interne gewählt und wird bei der Generierung LOAD-MODULE,...,ALTLIB=YES angegeben, so kann *open*UTM bei einem Programmaustausch das Modul nicht finden und meldet einen UNBIND-Fehler (`K078 UNBIND 0C010174`).

Mehrfach vorkommende Elemente

Beim Vorbinden von LLMs müssen Sie beachten, dass eingebundene Elemente (Sub-LLM / OM) nicht in mehreren, evtl. vorgebundenen LLM / OM vorkommen. Falls dies doch der Fall ist, müssen Sie die externen Namen, die in mehrfach eingebundenen Modulen enthalten sind, durch die BINDER-Anweisung MODIFY-SYMBOL-VISIBILITY maskieren.

Autolink-Funktion

Die Autolink-Funktion wird in der LOAD-MODULE-Anweisung durch den Operanden ALTERNATE-LIBRARIES gesteuert.

- ALTERNATE-LIBRARIES=NO
  schaltet die Autolink-Funktion des BLS aus. D.h. für alle dynamisch generierten Lademodule, die auf diese Weise geladen werden, müssen alle offenen Verweise eines solchen Lademoduls durch die zum Ladezeitpunkt geladenen Module (Start-LLM und andere Lademodule) und durch den Shared Code befriedigt werden. Externverweise ins Laufzeitsystem werden bei Lademodulen, die nur aus C- oder Datenobjekten bestehen, immer dann befriedigt, wenn beim Binden des Start-LLM ein RESOLVE-BY-AUTOLINK auf die Bibliothek SYSLNK.CRTE.PARTIAL-BIND angegeben wurde.
- ALTERNATE-LIBRARIES=YES
  schaltet die Autolink-Funktion des BLS ein. Diese können Sie für solche Lademodule verwenden, die beim Austausch weitere Module des Laufzeitsystems benötigen, die sich noch nicht im Speicher befinden. Beim Binden werden offene Externverweise zunächst im Link-Kontext und dann im Shared Code gesucht (siehe auch Handbuch „Bindelader - Starter“). Wenn danach Externverweise unbefriedigt geblieben sind, wird die Bibliothek durchsucht, die Sie im LIB-Operanden angegeben haben. Wird eine passende Definition gefunden, dann wird das zugehörige Modul eingebunden und die Suche abgebrochen. Ansonsten wird in den Bibliotheken weitergesucht, die Sie *open*UTM vor dem Start der Anwendung mit einem SET-FILE-LINK-Kommando über die Linknamen BLSLIB*nn* (wobei 00≤*nn*≤99), bekanntgemacht haben. Die Bibliotheken werden in aufsteigender Reihenfolge der nn bearbeitet.

  Die Behandlung der Module des Laufzeitsystems in Verbindung mit UTM-Anwendungen wird ausführlich auf Seite 46 beschrieben.

Der Autolink-Mechanismus sollte nicht auf benutzereigene Bibliotheken angewandt werden, da bei einem Programmaustausch nur das Lademodul und nicht die ganze Ladeeinheit, die zusätzlich alle durch Autolink geladenen Module enthält, entladen wird.

Beispiel

- UTM-Generierung und LINK-Anweisung

  UTM-Generierung:

```
LOAD-MODULE     lm-name                              -
                ,VERSION              = 001          -
                ,LIB                  = lm-lib       -
                ,LOAD-MODE            = ONCALL       -
                ,ALTERNATE-LIBRARIES = YES
```

  LINK-Anweisung vor Start der Anwendung:

```
/SET-FILE-LINK   LINK-NAME = BLSLIB00           -
/               ,FILE-NAME = $userid.SYSLNK.CRTE.PARTIAL-BIND
```

  Zunächst wird mit der LOAD-MODULE-Anweisung das Lademodul *lm-name* mit LOAD-MODE=ONCALL generiert, d.h. *lm-name* wird erst beim Aufruf eines darin enthaltenen Programms nachgeladen. Im Beispiel sollte *lm-name* COBOL85-Objekte enthalten.

  Um offene Externverweise beim Laden zu befriedigen, wird erst die im LIB-Operanden angegebene Bibliothek *lm-lib* durchsucht und dann die Bibliothek $*userid*.SYSLNK.CRTE.PARTIAL-BIND, die über das SET-FILE-LINK-Kommando zugewiesen ist.

- BINDER-Anweisungen

```
/START-BINDER
//START-LLM-CREATION     INTERNAL-NAME     = llm-name          -

//                      ,INTERNAL-VERSION = llm-version
//INCLUDE-MODULES         LIBRARY=benutzerbibliothek            -
//                      ,ELEMENT=teilprogrammname
//RESOLVE-BY-AUTOLINK     LIBRARY=(bibliothek1,bibliothek2..)
//SET-EXTERN-RESOLUTION   SYMBOL-TYPE=*REFERENCES               -
//                      ,RESOLUTION=*STD
//SAVE-LLM                LIBRARY=llm-bibliothek                -
//                      ,ELEMENT=*INTERNAL-NAME(                -
//                       VERSION=*INTERNAL-VERSION)             -
//                      ,FOR-BS2000-VERSIONS=*FROM-OSD-V4
//END
```

  In diesem BINDER-Lauf wird ein Teilprogramm der Anwendung zu einem Lademodul gebunden. KDCROOT wird hier nicht mit eingebunden, weswegen u.a. der Entry KDCS unbefriedigt bleibt.

3.2.3 LLMs mit Public/Private Slice binden

Wenn Sie mit dem Binder shareable Module im OM-Format zu LLMs mit einer Public Slice binden, dann müssen Sie darauf achten, dass die CSECTs dieser Module das Attribut PUBLIC besitzen. Ist dies nicht der Fall, legt der Binder diese CSECTs in die Private Slice.

Auftreten kann dies zum Beispiel bei AREAs, die vorher in shareable Modulen im OM-Format gewesen sind (hier hatte das Attribut keine Bedeutung), oder bei allen shareable COBOL85-Objekten. Sie können die Objekte anpassen, indem Sie

- bei AREAs das Attribut PUBLIC hinzufügen und dann neu übersetzen,
- bei shareable COBOL85-Objekten die zugehörigen Sourcen im LLM-Format neu übersetzen
- oder das PUBLIC-Attribut mit dem Binder setzen.

Das folgende Beispiel zeigt die dritte Möglichkeit.

Beispiel

Ein Lademodul soll als LLM vorgebunden werden. Es soll bestehen aus

- den COBOL85-Objekten AFPUT und COBECHO mit den nicht-shareable Teilen AFPUT und COBECHO und den shareable Teilen AFPUT@ und COBECHO@
- sowie den AREAs AREA1, AREA2 und AREA3, denen allen das PUBLIC-Attribut fehlt.

Dann ergeben sich folgende Binder-Anweisungen:

```
//START-LLM-CREATION                                                    -
//INTERNAL-NAME  = EXAMPLE-LLM                                          -
  //              ,INTERNAL-VERSION    = 001                            -
  //              ,SLICE-DEFINITION    = BY-ATTR( PUBLIC = YES )
//REMARK ------------------------------------------------------------------
//REMARK ------------------------------------------------------------------
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                    -
  //          ,ELE = COBECHO
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                    -
  //          ,ELE = AFPUT
//REMARK ------------------------------------------------------------------
//REMARK ------------------------------------------------------------------
//BEGIN-SUB-LLM SUB-LLM-NAME=OM-WITHOUT-PUBLIC-ATTRIBUTE
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                    -
  //          ,ELE = AFPUT@
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                    -
  //          ,ELE = COBECHO@
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                    -
  //          ,ELE = AREA1
```

```
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                    -
  //          ,ELE = AREA2
//INCLUDE-MOD  LIB = LIB.UTM.PRELINK                                    -
  //          ,ELE = AREA3
//MODIFY-SYMBOL-ATTR   PUBLIC = YES
//END-SUB-LLM
//REMARK ------------------------------------------------------------------
//REMARK ------------------------------------------------------------------
//SET-EXTERN-RESOLUTION  SYMBOL-TYPE=REFERENCE , RESOLUTION=STD
//MODIFY-MAP-DEFAULT  PROGRAM-MAP = PAR( DEFINITION = ALL               -
  //                                    ,INVERT         = ALL           -
  //                                    ,REFERENC       = ALL)          -
  //                                ,UNRESOLVED     = YES               -
  //                                ,SORTED-PRO     = YES               -
  //                                ,DUPLICATE      = YES               -
  //                                ,OUTPUT= LIST.LINK.EXAMPLE-LLM
/REMARK ------------------------------------------------------------------
//SAVE-LLM   LIB = LIB.LOAD-MODULE.STARTUP                              -
  //              ,ELEM                 = *INTERNAL                     -
  //              ,FOR-BS2000-VERSION  = *FROM-OSD-V4
//REMARK ------------------------------------------------------------------
//END
```

Wie das Beispiel zeigt, reicht es, alle shareable Module, die nicht das PUBLIC-Attribut besitzen, in dem Binder-Lauf in ein SUB-LLM zu binden, und dann für das ganze SUB-LLM das Attribut PUBLIC mit der Anweisung MODIFY-SYMBOL-ATTRIBUTES zu setzen. In der Liste LIST.LINK.EXAMPLE-LLM - Abschnitt *PROGRAM MAP* - kann man sich dann anschauen, in welchem Slice sich die einzelnen Objekte befinden.

3.2.4 Laufzeitsysteme binden

Viele Laufzeitsysteme besitzen Teile, die shareable sind. Dies kann zusammen mit BLS dazu benutzt werden, Ladezeiten zu verkürzen und Performance zu gewinnen. Dabei gibt es je nach Laufzeitsystem die folgenden Möglichkeiten:

- Shareable Teile als Subsystem laden. Dies ist die beste Möglichkeit, die aber nicht für alle Laufzeitsysteme möglich ist.
- Shareable Teile vorbinden und in einen Common Memory Pool laden.
- Die benötigten Teile des Laufzeitsystems zu einem LLM binden und dann dynamisch laden.
- Das Laufzeitsystem statisch zum Start-LLM binden.

Beim Binden des Start-LLMs müssen für *open*UTM das C- und SPL-Laufzeitsystem sowie das ILCS eingebunden werden (ILCS ist in den CRTE-Bibliotheken vorhanden). Dies erreichen Sie, indem Sie die beiden folgenden Binderanweisungen in dieser Reihenfolge geben:

```
//RESOLVE-BY-AUTOLINK LIB=$userid.SYSLNK.CRTE[.PARTIAL-BIND]
//RESOLVE-BY-AUTOLINK LIB=$userid.SYSLNK.UTM.052.SPLRTS
```

Wenn Sie Ihre Laufzeitsysteme statisch binden wollen, reichen diese beiden Anweisungen für C-, COBOL- und SPL-Objekte aus. Wenn Sie weitere Laufzeitsysteme benötigen, dann müssen Sie - sofern nichts anderes im zugehörigen Handbuch steht - hinter die erste RESOLVE-BY-AUTOLINK-Anweisung die RESOLVE-BY-AUTOLINK-Anweisungen für die benötigten Laufzeitsysteme einfügen.

Das Binden mit der SYSLNK.CRTE.PARTIAL-BIND hat den Vorteil, dass Binde- und Ladezeit verkürzt werden und dass das gebundene Programm weniger Plattenspeicher belegt. Außerdem bietet es Vorteile beim Austausch von Lademodulen, die von C-Sourcen stammen. Falls Sie SYSLNK.CRTE oder SYSLNK.CRTE.PARTIAL-BIND - neben dem Binden des Start-LLM - ein weiteres Mal benötigen, dann müssen Sie die jeweilige CRTE-Bibliothek nochmals verwenden. Ein solcher Fall ist das Binden des tasklokalen Teils des COBOL85-Laufzeitsystems, siehe Beispiel auf Seite 48.

Die Möglichkeiten für das shareable Laden werden in den Folgeabschnitten erläutert. Informationen darüber, welche Laufzeitroutinen shareable und welche nicht-shareable sind, können Sie der Beschreibung des jeweiligen Laufzeitsystems entnehmen.

Beachten Sie bitte, dass Sie ein Lademodul, das Module der Laufzeitsysteme enthält, nie während des laufenden Anwendungsprogramms austauschen dürfen, da dies zu schlecht diagnostizierbaren Fehlern führen kann.

3.2.4.1 Shareable LZS-Teile als Subsystem

Wenn Ihre UTM-Anwendung mit der gemeinsamen Laufzeitumgebung CRTE (Common Run Time Environment) abläuft, so besteht die Möglichkeit, die shareable Teile von CRTE über DSSM als Subsystem zu laden. Auch andere Laufzeitsysteme bieten diese Möglichkeit an.

Das Laden des Subsystems ist Aufgabe des BS2000-Systemverwalters, näheres hierzu ist im Handbuch CRTE beschrieben.

Wenn möglich, sollten Sie von diesem Verfahren Gebrauch machen, da es Systemressourcen schont. Das Laufzeitsystem für C, das in CRTE enthalten ist, sollte jedoch unbedingt als Subsystem geladen werden, da es vom KDCROOT benötigt wird. Das Vorgehen hängt davon ab, ob das C- oder das COBOL-Laufzeitsystem vorgeladen ist:

- Wenn das C-Laufzeitsystem vorgeladen ist, dann können Sie dieses nur nutzen, falls Sie die Bibliothek SYSLNK.CRTE.PARTIAL-BIND einbinden. Die Verbindung zum C-Subsystem wird dann durch ein Verbindungsmodul hergestellt, das beim Binden des Start-LLM eingebunden wird. Ist das C-Laufzeitsystem nicht erreichbar, dann lädt das Verbindungsmodul die notwendigen Teile aus der CRTE-Bibliothek in den tasklokalen Teil nach.
- Wenn das COBOL-Laufzeitsystem vorgeladen ist, dann binden Sie die benötigten nicht-shareable Teile des Laufzeitsystems zu einem Lademodul zusammen, siehe Beispiel auf Seite 48.

3.2.4.2 Shareable LZS-Teile in Common Memory Pools

Wenn Module ein Laufzeitsystem benutzen, das nicht als Subsystem geladen werden kann und das auch nicht im Start-LLM eingebunden ist, dann sollten Sie alle shareable Module des Laufzeitsystems jeweils einer Programmiersprache funktionsgruppenweise zu LLMs vorbinden und durch *open*UTM in einen Common Memory Pool laden. Zum Laden der nicht-shareable Teile des Laufzeitsystems sind beide im vorherigen Abschnitt geschilderten Varianten möglich.

Die shareable Teile des Laufzeitsystems müssen Sie dann wie folgt generieren:

```
LOAD-MODULE share-rts-part,LOAD-MODE=(POOL,..., NO-PRIVATE-SLICE),...
```

Zudem haben Sie noch die Möglichkeit, die shareable und die nicht-shareable Teile zu einem LLM zu binden. Dieses Lademodul müssen Sie generieren mit:

```
LOAD-MODULE rts,LOAD-MODE=(POOL,..., STARTUP),...
```

Beim Generieren müssen Sie darauf achten, dass dieses LLM **vor** den anderen Modulen angegeben wird, die das Laufzeitsystem benötigt, da es sonst zu unbefriedigten Externverweisen führt.
Beachten Sie bitte, dass shareable Teile des C-, des SPL- und des FOR1-Laufzeitsystems nicht in einen von *open*UTM verwalteten Common Memory Pool geladen werden dürfen.

3.2.4.3 Laufzeitsysteme zu einem LLM binden

Für alle Laufzeitsysteme, die keine PARTIAL-BIND-Bibliothek besitzen, müssen Sie feststellen, welche Module des Laufzeitsystems für die gesamte Anwendung benötigt werden. Dazu gehen Sie folgendermaßen vor:

1. Alle Anwendungsteilprogramme und alle Event-Exits der gewünschten Sprache sowie alle shareable Module des Laufzeitsystems, die als Subsystem oder in einen Common Memory Pool geladen sind, müssen zu einem Sub-LLM zusammengebunden werden.
2. Das ILCS (enthalten in CRTE) muss ebenfalls in das Sub-LLM eingebunden werden, und zwar die höchste zur Verfügung stehende Version.
3. Um alle Externverweise auf das Laufzeitsystem zu befriedigen, muss die RESOLVE-BY-AUTOLINK-Anweisung auf die Bibliothek des Laufzeitsystems der Sprache eingegeben werden: RESOLVE-BY-AUTOLINK LIBRARY=*rts-bibliothek*.
4. Mit Hilfe der Anweisung REMOVE-MODULES müssen Sie dann das Sub-LLM (siehe 1. und 2.) wieder aus dem LLM entfernen.
5. Das Lademodul generieren Sie in *open*UTM mit folgender Anweisung:

```
LOAD-MODULE COB85-RTS-PRIVATE-KERNEL    -
            ,VERSION=001                -
            ,LIB=private-rts-lib        -
            ,LOAD-MODE=STARTUP
```

Beispiel

Der nachfolgende BINDER-Lauf zeigt, wie Sie ein LLM aus den Modulen des Laufzeitsystems für COBOL85 binden können. Es wird also vorausgesetzt, dass die Anwendungsteilprogramme in COBOL85 geschrieben sind. Außerdem wird angenommen, dass der Shared Code des Laufzeitsystems für COBOL85 als Subsystem geladen ist, dass im Start-LLM keine weiteren COBOL85-Module enthalten sind und dass keine Module des C-Laufzeitsystems benötigt werden; z.B. benutzt COBOL85 V2.2 Module des C-Laufzeitsystems, wenn die Option COMOPT ENABLE-UFS-ACCESS=YES gesetzt ist.

Im Beispiel wird von CRTE V2.3 und Installation über IMON ausgegangen.

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=COB85-RTS-PRIVATE-KERNEL      -
                    ,INTERNAL-VERSION=001
//REMARK +-------------------------------------------------------+
//BEGIN-SUB-LLM-STATEMENTS SUB-LLM-NAME=COB85-LM-SHARED-RTS-ILCS  ———— 1.
//INCLUDE-MODULES ELEMENT=oncall-load-module
//INCLUDE-MODULES ELEMENT=startup-load-module
//INCLUDE-MODULES ELEMENT=pool-load-module
//INCLUDE-MODULES LIBRARY=$userid.SYSLNK.CRTE.PARTIAL-BIND          -
                  ,ELEMENT=(ITOSL@, ITOSL#)  ——————————————————————  2.
```

```
//INCLUDE-MODULES LIBRARY=$userid.SYSLNK.CRTE.SHARE                    -
                 ,ELEMENT=ITCSR85
//END-SUB-LLM-STATEMENTS
//REMARK  +-----------------------------------------------------------+
//RESOLVE-BY-AUTOLINK LIBRARY=$userid.SYSLNK.CRTE.PARTIAL-BIND    ---      3.
//REMARK  +-----------------------------------------------------------+
//SHOW-MAP ..., UNRESOLVED-LIST=SORTED, ...  ------------------------------  4.
//REMARK  +-----------------------------------------------------------+
//REMOVE-MODULES NAME=*ALL,PATH-NAME=.COB85-LM-SHARED-RTS-ILCS  -------  5.
//REMARK  +-----------------------------------------------------------+
//SET-EXTERN-RESOLUTION SYMBOL-TYP=*REFERENCES,RESOLUTION=*STD
//REMARK  +-----------------------------------------------------------+
//SAVE-LLM LIBRARY=private-rts-lib                                     -
//          ,ELEMENT=*INTERNAL-NAME(VERSION=*INTERNAL-VERSION)       -
//          ,FOR-BS2000-VERSIONS=*FROM-OSD-V4
//END
```

1. Zunächst wird ein Sub-LLM definiert, in das die POOL, STARTUP oder ONCALL generierten Lademodule mit den Teilprogrammen der Anwendung eingebunden werden.
2. Ebenso werden in dieses Sub-LLM das ILCS (Module mit Präfix IT0) und alle shareable Teile der benötigten Laufzeitsysteme (Großmodul ITCSR85) eingebunden. Das ILCS muss schon im Start-LLM enthalten sein; die shareable Module sind entweder in Common Memory Pools oder als Subsystem verfügbar. Dieses Sub-LLM soll ja vor dem Abspeichern des LLM wieder entfernt werden (siehe 5.).
3. Mit der RESOLVE-BY-AUTOLINK-Anweisung werden die übrigen noch benötigten Module der Laufzeitsysteme eingebunden.
4. Mit der SHOW-MAP-Anweisung können Sie sich u.a. die nicht aufgelösten Externverweise auflisten lassen. Die Liste, die an dieser Stelle des BINDER-Laufs erstellt wird, enthält alle offenen Externverweise, die noch befriedigt werden müssen. Alle Module des Laufzeitsystem, die benötigt werden, sollten im Lademodul enthalten sein, d.h. es sollten keine offenen Externverweise auf das Laufzeitsystem von COBOL85 oder auf ILCS-Module existieren.

   Auch am Ende des BINDER-Laufs können Sie eine Liste der unbefriedigten Externverweise erhalten, wenn Sie vor der Anweisung SAVE-LLM die Anweisung MODIFY-MAP-DEFAULTS mit der Option UNRESOLVED-LIST=SORTED angeben. Diese Liste enthält aber zusätzlich alle die unbefriedigten Externverweise, die durch das Entfernen des Sub-LLM entstehen.
5. Vor dem Abspeichern des LLM wird das zunächst gebundene Sub-LLM wieder entladen, so dass nur die über RESOLVE-BY-AUTOLINK gebundenen Module im LLM verbleiben. Wie im vorigen Beispiel dürfen in der SET-EXTERN-RESOLUTION-Anweisung die Standardwerte nicht verändert werden.

3.2.5 Start-LLM binden

Beim Binden des Start-LLM wird zwischen den folgenden Möglichkeiten unterschieden:

- Sie binden das ROOT-Tabellenmodul statisch in das Start-LLM mit ein oder
- das ROOT-Tabellenmodul wird dynamisch beim Start des Anwendungsprogramms nachgeladen. Dazu müssen Sie in den Startparametern für TABLIB den Namen der Bibliothek, die das ROOT-Tabellenmodul enthält, und für ROOTNAME den Namen des ROOT-Tabellenmoduls (PLAM-Elementname) angeben (siehe Abschnitt „Startparameter für openUTM“ auf Seite 82). Die Module KDCRTST, KDCRTCF und KDCRTCU müssen Sie statisch einbinden, und zwar das Modul KDCRTST als erstes.

Das zweite Verfahren ist vorteilhafter, da Sie bei Änderungen in der Generierung die Anwendung nicht neu binden müssen. Aus dem gleichen Grund sollten Sie möglichst alle Teilprogramme in Lademodule ablegen, die dynamisch gebunden werden.

Bei beiden Verfahren müssen Sie die folgenden Punkte beachten:

- Die Laufzeitsysteme für C und SPL zusammen mit dem ILCS werden von den Modulen des KDCROOT benötigt, weswegen Sie die nicht-shareable Teile des Laufzeitsystems immer statisch einbinden müssen.
- Das Start-LLM darf nach dem Binden nur unbefriedigte Externverweise auf solche Module der Laufzeitsysteme haben, die entweder vor dem Anwendungsstart als Subsystem geladen sind oder die beim Anwendungsstart nachgeladen werden.
- Sie sollten vermeiden, mehrere Start-LLM in der gleichen Bibliothek abzulegen, da BLS beim Starten versucht, offene Externverweise aus dieser Bibliothek zu befriedigen, d.h. neben Lademodulen, die dynamisch nachgeladen werden, darf die Bibliothek nur ein Start-LLM enthalten, das mit /START-PROGRAM geladen wird.
- Die RESOLVE-Anweisung auf die Bibliothek SYSLNK.CRTE.PARTIAL-BIND bzw. SYSLNK.CRTE müssen Sie stets als erste RESOLVE-Anweisung auf eine Laufzeitsystem-Bibliothek und nach der RESOLVE-Anweisung auf die UTM-Bibliothek angeben. Die Bibliothek, die Sie im Start-LLM zum Binden der benötigten Module des CRTE verwenden, müssen Sie auch beim Binden aller Lademodule der Anwendung angeben. Die Bibliothek SYSLNK.CRTE.PARTIAL-BIND bringt Performancevorteile und sollte daher bevorzugt eingesetzt werden.
- Wird beim Erzeugen des Start-LLM bei der Anweisung START-LLM-CREATION der Parameter SLICE-DEFINITION=BY-ATT verwendet, so kann es vorkommen, dass der Binder die physikalische Struktur der ROOT-Systemmodule unerwünscht verändert. Um dies zu vermeiden, sollten die Module KDCRTST, KDCRTCF und KDCRTCU als TYPE=L eingebunden werden.

Beispiel

Der folgende BINDER-Lauf enthält alle Anweisungen, die zum Binden eines Start-LLM benötigt werden. Es wird davon ausgegangen, dass die Anwendung mit der gemeinsamen Laufzeitumgebung CRTE V2.3 abläuft, und über IMON installiert wurde:

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=start-llm                              -
//                  ,INTERNAL-VERSION=start-llm-version
//REMARK  +-------------------------------------------------------------+
//BEGIN-SUB-LLM-STATEMENTS SUB-LLM-NAME=ROOT-TAB-LLM ------------------- 1.
//INCLUDE-MODULES LIBRARY=tablib ,ELEMENT=root-module
//END-SUB-LLM-STATEMENTS
//REMARK  +-------------------------------------------------------------+
//INCLUDE-MODULES LIBRARY=$userid1.SYSLNK.UTM.052                         -
//               ,ELEMENT=KDCRTST  ------------------------------------- 2.
//INCLUDE-MODULES LIBRARY=$userid1.SYSLNK.UTM.052                         -
//               ,ELEMENT=KDCRTCF
//INCLUDE-MODULES LIBRARY=$userid1.SYSLNK.UTM.052                         -
//               ,ELEMENT=KDCRTCU
//REMARK  +-------------------------------------------------------------+
//BEGIN-SUB-LLM-STATEMENTS SUB-LLM-NAME=LM-SHARED-RTS ------------------ 3.
//INCLUDE-MODULES LIBRARY=$userid2.SYSLNK.CRTE.SHARE                      -
//               ,ELEMENT=ITCSR85
//INCLUDE-MODULES oncall-load-module
//INCLUDE-MODULES startup-load-module
//INCLUDE-MODULES pool-load-module
//END-SUB-LLM-STATEMENTS
//REMARK  +-------------------------------------------------------------+
//RESOLVE-BY-AUTOLINK LIBRARY=$userid1.SYSLNK.UTM.052 ------------------ 4.
//RESOLVE-BY-AUTOLINK LIBRARY=user-lib
//REMARK  +-------------------------------------------------------------+
//RESOLVE-BY-AUTOLINK LIBRARY=$userid2.SYSLNK.CRTE.PARTIAL-BIND -------- 5.
//RESOLVE-BY-AUTOLINK LIBRARY=other-rts-lib
//RESOLVE-BY-AUTOLINK LIBRARY=$userid1.SYSLNK.UTM.052.SPLRTS
//REMARK  +-------------------------------------------------------------+
//SHOW-MAP ..., UNRESOLVED-LIST=SORTED, ... ---------------------------- 6.
//REMARK  +-------------------------------------------------------------+
//REMOVE-MODULES NAME=*ALL,PATH-NAME=.ROOT-TAB-LLM --------------------- 7.
//REMOVE-MODULES NAME=*ALL,PATH-NAME=.LM-SHARED-RTS -------------------- 8.
//REMARK  +-------------------------------------------------------------+
//SET-EXTERN-RESOLUTION SYMBOL-TYP=*REFERENCES,RESOLUTION=*STD
//SAVE-LLM LIBRARY=start-bibliothek,ELEMENT=*INTERNAL-NAME(               -
//                                  VERSION=*INTERNAL-VERSION)            -
//               ,FOR-BS2000-VERSION=*FROM-OSD-V4
//END
```

1. Als erstes wird immer das ROOT-Tabellenmodul in ein eigenes Sub-LLM gebunden. Dieses Modul können Sie vor dem Abspeichern des Start-LLM leicht wieder entfernen (siehe 7), falls das ROOT-Tabellenmodul, wie hier im Beispiel, nicht statisch eingebunden, sondern beim Start des Anwendungsprogramms dynamisch nachgeladen werden soll.

   Durch die Externverweise des ROOT-Tabellenmoduls werden die statischen Teilprogramme aus *user-lib* dazugebunden (siehe 4). Dies sollten allerdings möglichst wenige Teilprogramme sein, da diese Module nicht ausgetauscht werden können.

2. Als nächstes werden die ROOT-Systemmodule KDCRTST, KDCRTCF und KDCRTCU gebunden. Diese müssen immer dann explizit eingebunden werden, wenn ein Start-LLM ohne ROOT-Tabellenmodul erzeugt werden soll.

3. In ein eigenes Sub-LLM wird der shareable Teil des COBOL85-Laufzeitsystems gebunden. Der shareable Teil des COBOL85-Laufzeitsystems muss vor dem Start der Anwendung vom Administrator als eigenständiges Subsystem oder von *open*UTM bei Anwendungsstart in einen Common Memory Pool geladen werden.

   Zusätzlich werden alle vorhandenen Lademodule, die dynamisch nachgeladen werden, in dieses Sub-LLM gebunden, damit die benötigten Module des Laufzeitsystems durch die RESOLVE-BY-AUTOLINK-Anweisung zum Start-LLM hinzugebunden werden. Dadurch sind sie beim dynamischen Nachladen schon vorhanden und brauchen nicht mehr über die Autolink-Funktion nachgeladen werden, weil dies die Ladeperformance verschlechtert.

   Bei den STARTUP-Lademodulen sollte auch ein Lademodul mit dem benötigten nicht-shareable Teil des COBOL85-Laufzeitsystems dabei sein. Lädt man den shareable Teil des COBOL85-Laufzeitsystems (ITCSR85) in einen von *open*UTM verwalteten Common Memory Pool, dann muss ein solches nicht-shareable Lademodul vorhanden sein (z. B. COB85-RTS-PRIVATE-KERNEL im Beispiel auf Seite 48). Andernfalls können die „unresolved externs“ auf den nicht-shareable Teil des COBOL85-Laufzeitsystems nicht aufgelöst werden, weil sich die nicht-shareable Teile des COBOL85-Laufzeitsystems dann im Start-LLM befinden.

   Benutzt man das COBOL85-Subsystem, dann benötigt man das STARTUP-Lademodul nicht, da die „unresolved externs“ auf die shareable Teile des Laufzeitsystems beim Start aufgelöst werden können.

4. Als erste Bibliothek in einer RESOLVE-BY-AUTOLINK-Anweisung müssen Sie stets die UTM-Bibliothek angeben. Danach können RESOLVE-BY-AUTOLINK-Anweisungen auf Benutzerbibliotheken folgen.

5. Als erste Bibliothek eines Laufzeitsystems in einer RESOLVE-BY-AUTOLINK-Anweisung müssen Sie stets die CRTE-Bibliothek angeben. Nach der CRTE-Bibliothek sollten Sie die RESOLVEs auf eventuell vorhandene Bibliotheken anderer Laufzeitsysteme eingeben, und zwar vor der Bibliothek mit dem von *open*UTM benötigten SPL-Laufzeitsystem. Weitergehende Informationen sind in dem Handbuch des entsprechenden Laufzeitsystems zu finden.
6. Mit der SHOW-MAP-Anweisung können Sie sich u.a. die nicht aufgelösten Externverweise auflisten lassen. Die Liste, die an dieser Stelle des BINDER-Laufs erstellt wird, enthält alle offenen Externverweise, die noch befriedigt werden müssen. Am Ende des BINDER-Laufs erhalten Sie automatisch eine weitere Liste der unbefriedigten Externverweise, die aber zusätzlich alle die unbefriedigten Externverweise enthält, die durch das Entfernen des Sub-LLM entstehen.
7. Mit einer REMOVE-MODULES-Anweisung entfernen Sie das ROOT-Tabellenmodul aus dem Start-LLM.
8. Auch die shareable Module des Laufzeitsystems für COBOL85 und die Lademodule, die mit ONCALL, STARTUP oder POOL generiert sind, werden aus dem Start-LLM entfernt.

Das so gebundene Start-LLM muss mit dem folgenden Kommando geladen werden:

```
/START-PROGRAM FROM-FILE=*MODULE                                    -
                     (LIBRARY=start-bibliothek                      -
                     ,ELEMENT=start-llm                             -
                     ,RUN-MODE=*ADVANCED                            -
                               (UNRESOLVED-EXTRNS=*DELAY            -
                               ,LOAD-INFORMATION=*REFERENCES)       -
                     ,PROGRAM-MODE=24/ANY) )
```

3.3 Hinweise für Anwendungen mit ILCS-Teilprogrammen

- Programmübergänge (CALL Aufrufe) zwischen 'ILCS'-Teil- oder -Unterprogrammen und 'Nicht-ILCS'-Unterprogrammen sind verboten.
- ILCS-Module (Präfix IT0....) dürfen im geladenen Anwendungsprogramm nur einmal enthalten sein! D.h. Sie müssen verhindern, dass die ILCS-Module beim Binden per RESOLVE aus den RTS-Bibliotheken der Compiler mehrfach hinzugebunden werden oder dass sie durch dynamisches Nachladen von Modulen mehrfach auftreten.
- Es muss immer die höchste zur Verfügung stehende ILCS-Version geladen werden. In der CRTE-Bibliothek ist immer ein ILCS enthalten.
- Ob die betreffende Compilerversion bzw. das Laufzeitsystem ILCS unterstützt und welche ILCS-Version, ist dem Benutzerhandbuch bzw. Serviceinformationen wie Freigabemitteilung und Readme-Dateien des Compilers zu entnehmen.

  Sie finden auf Seite 245ff eine detaillierte Aufstellung mit den möglichen Kombinationen von Compiler-Option und RTS.
- Beispiele zum Binden von ILCS-Programmen sind im CRTE-Handbuch zu finden.

3.4 Änderungen im Anwendungsprogramm bei der Umstellung auf die BLS-Funktionen

Wenn Sie eine Anwendung auf die Nutzung von BINDER und BLS umstellen wollen, müssen Sie die folgenden Schritte durchführen:

1. Sie sollten die Struktur des Anwendungsprogramms überarbeiten, um es sinnvoll in statisch zu bindende und nachzuladende Programmteile aufzuteilen. Die nachzuladenden Programmteile sollten Sie sinnvoll auf einzelne Lademodule aufteilen, Siehe dazu auch Kapitel „Struktur des Anwendungsprogramms festlegen“ auf Seite 13.
2. Die Bindeprozeduren müssen Sie der neuen Anwendungsstruktur anpassen und auf die Nutzung von BINDER umstellen.
3. Sie müssen die Generierung auf die neuen Anweisungen und Operanden umstellen Für jedes Lademodul (LLM) oder Objektmodul (OM) muss eine LOAD-MODULE-Anweisung generiert werden. Die PROGRAM- und AREA-Anweisungen müssen über den Operanden LOAD-MODULE den Lademodulen zugeordnet werden. Die alten Anweisungen MODULE und ENTRY sowie die Operanden LOAD müssen entfernt werden. Mehr Informationen dazu finden Sie im *open*UTM-Handbuch „Anwendungen generieren“.
4. Das vorgebundene Anwendungsprogramm müssen Sie in einer Programmbibliothek bereitstellen.
5. Die Startprozedur müssen Sie auf die von BLS unterstützte Syntax umstellen.

Bei der Umstellung von TSOSLNK/alte DLL-Schnittstelle auf BINDER/BLS müssen Sie die folgenden daraus resultierenden Änderungen beachten:

- Es kann zu Namenskollisionen kommen, da das BLS die Namen aller CSECTs, ENTRYs und COMMONs im gesamten Laderaum (CONTEXT) kennt. Das ist unabhängig davon, ob das Anwendungsprogramm ganz, teilweise oder gar nicht vorgebunden ist, d.h. ob das gesamte Anwendungsprogramm mit einer LOAD-PROGRAM-Anweisung gestartet wird, oder ob Teile des Anwendungsprogramms durch BIND-Aufrufe nachgeladen werden.
- Eine Referenz (z. B. eine V-Konstante) in einem Modul, das in einem Common Memory Pool liegt, auf eine (CSECT- oder ENTRY-)Definition eines Moduls, das sich in einem anderen Common Memory Pool befindet, kann bei Nutzung der BLS-Schnittstelle - im Gegensatz zur alten DLL-Schnittstelle - nicht aufgelöst werden.

4 Für den Betrieb notwendige Dateien

Vor jedem Starten einer UTM-Anwendung müssen Sie dafür sorgen, dass die folgenden, für den Betrieb einer UTM-Anwendung notwendigen Dateien vorhanden sind:

- die KDCFILE
- die System-Protokolldatei SYSLOG
- die Benutzer-Protokolldatei(en) USLOG (optional)
- alle Programm- und Objektmodul-Bibliotheken, aus denen beim Starten und während des Betriebs der Anwendung Module dynamisch nachgeladen werden sollen.

Den Basisnamen *filebase* für die benötigten Dateien legen Sie beim Erzeugen der KDCFILE mit dem Generierungstool KDCDEF fest, siehe *open*UTM-Handbuch „Anwendungen generieren“. Basisname der Dateien ist der bei der Generierung in der MAX...,KDCFILE=*filebase* festgelegte Basisname.

4.1 System-Protokolldatei SYSLOG

*open*UTM protokolliert alle Ereignisse aus dem Lauf der Anwendung in die System-Protokolldatei SYSLOG (**SYS**TEM **LOG**GING), d.h. *open*UTM schreibt alle Meldungen mit Meldungsziel SYSLOG in diese Datei (zu „Meldungsziel“ siehe *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“). Dabei arbeitet *open*UTM mit Wechselpuffern. Dies vermeidet Wartesituationen und verbessert so die Performance, vor allem in Anwendungen mit einem hohen Aufkommen an SYSLOG-Meldungen.

Die System-Protokolldatei SYSLOG können Sie für die laufende Überwachung des Anwendungslaufs oder für spätere Kontrollen nutzen. Insbesondere für die Diagnose liefert die SYSLOG wichtige Information.

Pro Anwendung müssen Sie *open*UTM eine SYSLOG zur Verfügung stellen. Wenn Sie *open*UTM vor dem Start keine SYSLOG zugewiesen haben, bricht *open*UTM den Start der Anwendung ab.

Die System-Protokolldatei SYSLOG können Sie anlegen als:

- einfache Datei SYSLOG
- Dateigenerationsgruppe SYSLOG-FGG (**F**ile **G**eneration **G**roup)

Gegenüber einer einfachen SYSLOG-Datei hat eine SYSLOG-FGG folgende Vorteile:

- Sie können im laufenden Betrieb der Anwendung auf die jeweils folgende Dateigeneration umschalten (umschaltbare SYSLOG-Datei). Die SYSLOG können Sie z.B. mit dem Administrationskommando KDCSLOG administrieren. Sehen Sie dazu *open*UTM-Handbuch „Anwendungen administrieren". Wenn alle Tasks der Anwendung die alte SYSLOG-Datei geschlossen haben, können Sie frei über diese Dateigeneration verfügen.
- Sie können eine automatische Größenüberwachung für die SYSLOG einstellen. D.h. Sie können einen Schwellwert für die Größe der einzelnen Dateigenerationen der SYSLOG-FGG generieren bzw. per Administration festlegen, bei dem *open*UTM automatisch auf die folgende Dateigeneration der FGG umschaltet. Die Größenüberwachung kann während des Anwendungsbetriebs ein- und ausgeschaltet werden.

Meldungen von *open*UTM

*open*UTM gibt die folgenden Meldungen bezüglich der SYSLOG aus:

- Beim Anwendungsstart die Meldung K136:

```
K136 (Erste) SYSLOG-Datei ist &FNAM
```

- Beim Anwendungsende oder nach Umschalten auf eine andere Dateigeneration die Meldung K138:

```
K138 SYSLOG-Datei &FNAM geschlossen
```

4.1.1 SYSLOG als einfache Datei

Wenn die SYSLOG-Datei als einfache Datei geführt werden soll, dann können Sie diese *open*UTM auf zwei Arten bekannt machen:

- Vor jedem Start der Anwendung müssen Sie der Datei den Link-Namen SYSLOG zuweisen (SET-FILE-LINK-Kommando). Beim Start der Anwendung öffnet *open*UTM diese Datei. Sie bleibt während des gesamten Anwendungslaufs für alle Tasks der Anwendung geöffnet.
- Sie können auch vor dem Start eine Datei mit dem Namen *filebase*.SLOG anlegen. Die Datei muss dieselbe Katalogkennung (CATID) haben wie die KDCFILE *filebase*.KDCA (siehe *open*UTM-Handbuch „Anwendungen generieren", Anweisung MAX...,CATID und KDCFILE). *open*UTM benutzt die Datei mit dem Namen *filebase*.SLOG bei jedem folgenden Start als SYSLOG-Datei, solange beim Start der Anwendung keine Datei bzw. Dateigeneration mit dem Link-Namen SYSLOG existiert. Ist eine Datei bzw. Dateigeneration mit dem Link-Namen SYSLOG vorhanden, dann protokolliert *open*UTM die SYSLOG-Meldungen immer in dieser Datei.

Nach dem Ende eines Anwendungslaufs sollten Sie den Inhalt der Datei *filebase*.SLOG sichern. Beim folgenden Start der Anwendung wird der Inhalt dieser Datei von *open*UTM überschrieben.



ACHTUNG!
Wollen Sie die SYSLOG als einfache Datei führen, dann dürfen Sie die Größenüberwachung für die SYSLOG **nicht** generieren. Wenn Sie in diesem Fall die Größenüberwachung bei der Generierung einschalten mit MAX...,SYSLOG-SIZE=*size* (*size* > 0), dann bricht *open*UTM den Start der Anwendung mit Startfehlercode 58 ab.

4.1.2 Dateigenerationsgruppe SYSLOG-FGG

Wenn die SYSLOG-Datei als FGG geführt wird, dann öffnet *open*UTM beim Start der Anwendung eine Dateigeneration der SYSLOG-FGG als SYSLOG-Datei. Alle Tasks der Anwendung schreiben die Meldungen mit Ziel SYSLOG zunächst in diese Dateigeneration.

Wird die SYSLOG als Dateigenerationsgruppe angelegt, dann reicht es, wenn Sie vor dem Start die Dateigenerationsgruppe (FGG) anlegen. Die einzelnen Dateigenerationen müssen Sie nicht anlegen, weil *open*UTM vor dem Öffnen einer SYSLOG-Dateigeneration prüft, ob diese schon existiert. Existiert die Dateigeneration, dann wird sie lediglich geöffnet. Existiert die Dateigeneration noch nicht, dann legt *open*UTM die Dateigeneration selbstständig an, mit den Merkmalen der Dateigenerationsgruppe.

Wollen Sie den Dateigenerationen bestimmte Werte für Primary und Secondary Allocation zuordnen, dann müssen Sie eine Dateigeneration der FGG mit den gewünschten Werten anlegen. *open*UTM legt dann die restlichen Dateigenerationen mit denselben Werten an.

SYSLOG-FGG bekanntmachen

Sie haben die folgenden Möglichkeiten *open*UTM die SYSLOG-FGG bekannt zu machen:

- Verwendung des Link-Namens "SYSLOG"

  Sie legen vor dem Start der Anwendung eine FGG mit beliebigem Namen für die SYSLOG an. Ordnen Sie dann einer Dateigeneration der FGG mit /SET-FILE-LINK den Link-Namen SYSLOG zu. Die Zuweisung des Link-Namens muss vor jedem Start der Anwendung erfolgen.

  In diesem Fall wird die Dateigenerationsgruppe, zu der die Dateigeneration mit dem Link-Namen SYSLOG gehört, als SYSLOG-FGG genommen. *open*UTM protokolliert zunächst in die Dateigeneration mit dem Link-Namen SYSLOG (erste SYSLOG-Dateigeneration).

Haben Sie für Ihre Anwendung die automatische Größenüberwachung generiert, dann schaltet *open*UTM auf die nächste Dateigeneration dieser FGG um, sobald die Größe der ersten SYSLOG-Dateigeneration den Schwellwert der Größenüberwachung erreicht hat. Existiert diese Dateigeneration noch nicht, legt *open*UTM sie selbst an.

Wurde die Anwendung ohne automatische Größenüberwachung generiert, dann protokolliert *open*UTM solange in die Dateigeneration mit dem Link-Namen SYSLOG, bis Sie auf eine andere Dateigeneration der FGG weiterschalten oder die automatische Größenüberwachung einschalten (z.B. mit dem Administrationskommando KDCSLOG). Diese Dateigeneration wird ebenfalls beim Umschalten von *open*UTM automatisch angelegt.

- Vor dem ersten Start der Anwendung eine FGG mit dem Namen *filebase*.SLOG anlegen

  Diese Dateigenerationsgruppe muss dieselbe Katalogkennung (CATID) haben wie die KDCFILE *filebase*.KDCA (siehe *open*UTM-Handbuch „Anwendungen generieren“, Anweisung MAX CATID und KDCFILE). *open*UTM benutzt die FGG mit dem Namen *filebase*.SLOG nur als SYSLOG-FGG, wenn beim Start der Anwendung keine Datei bzw. Dateigeneration unter der Benutzerkennung der Anwendung vorhanden ist, der der Link-Name SYSLOG zugeordnet ist. Ist eine Datei bzw. Dateigeneration mit dem Link-Namen SYSLOG vorhanden, dann protokolliert *open*UTM die SYSLOG-Meldungen immer in dieser Datei.

  Ist *filebase*.SLOG eine Dateigenerationsgruppe, dann entscheidet die eingestellte Basis der FGG, welche Dateigeneration als erste SYSLOG-Datei genommen wird.

  Basis außerhalb des gültigen Bereichs

  Liegt die Basis außerhalb des gültigen Bereichs (z.B. BASE-NUM=0), dann legt *open*UTM beim Start der Anwendung die Dateigeneration mit der Generationsnummer LAST-GEN+1 an. Diese Dateigeneration ist dann die erste SYSLOG-Datei.

  Beim nächsten Start der Anwendung nimmt *open*UTM als erste SYSLOG-Datei die Folgegeneration der im vorangegangenen Anwendungslauf zuletzt beschriebenen Dateigeneration. D.h. wurde im letzten Anwendungslauf bis zur *n*-ten Dateigeneration geschrieben, dann wird beim nächsten Start in der (*n*+1)-ten Dateigeneration begonnen.

  Basis innerhalb des gültigen Bereichs

  Liegt die Basis innerhalb des gültigen Bereichs zwischen der ersten und der letzten Dateigeneration (in der Ausgabe des Kommandos SHOW-FILE-ATTRIBUTES auf die FGG ist: FIRST-GEN ≤ BASE-NUM ≤ LAST-GEN), dann wird die Basisgeneration als erste SYSLOG-Datei genommen.

  *open*UTM verändert die von Ihnen eingestellte Basisnummer nicht. Beim nächsten Start der Anwendung beginnt *open*UTM mit der Protokollierung wieder in derselben Dateigeneration wie beim vorherigen Start, es sei denn, Sie haben zuvor die Basiseinstellung geändert.

4.1.2.1 SYSLOG-FGG anlegen

Wollen Sie mit einer SYSLOG-FGG arbeiten, dann müssen Sie diese vor dem Start der Anwendung anlegen.

SYSLOG als *filebase*.SLOG anlegen

Im einfachsten Fall arbeiten Sie mit der FGG *filebase*.SLOG, bei der die Basis außerhalb des gültigen Bereichs liegt. Diese FGG müssen Sie nur einmal vor dem ersten Start der Anwendung anlegen. Bei jedem weiteren Anwendungsstart würde *open*UTM automatisch mit der Folgegeneration zur zuletzt beschriebenen Dateigeneration weitermachen, sofern Sie die Basis nicht in den gültigen Bereich verlegen. Sie können die FGG mit dem folgenden BS2000-Kommando anlegen:

```
/CREATE-FILE-GROUP                                        –
/   GROUP-NAME = filebase.SLOG                            –
/  ,GENERATION-PARAMETER = GENERATION-PARAMETER(          –
/       MAXIMUM = n                                       –
/     [,VOLUME = volume                                   –
/      ,DEVICE-TYPE = device                              –
/      ,OVERFLOW-OPTION = overflow ] )
```

Bedeutung der Parameter:

MAXIMUM=n maximale Anzahl der Dateigenerationen, die in der FGG gleichzeitig katalogisiert sein dürfen.

VOLUME=volume, DEVICE=device
Datenträgerkennzeichen und Gerätetyp der Platte, auf der die FGG eingerichtet werden soll. Die FGG kann auf PUBLIC oder PRIVATE DISK eingerichtet werden.

OVERFLOW-OPTION=overflow
gibt an, was geschehen soll, wenn die maximal erlaubte Anzahl an Dateigenerationen (MAXIMUM) überschritten wird. Über diesen Operanden können Sie steuern, ob von der SYSLOG-FGG Ihrer Anwendung nur die letzten *n* (MAXIMUM) Dateigenerationen erhalten bleiben sollen oder alle von *open*UTM beschriebenen Dateigenerationen. Sie können CYCLIC-REPLACE, REUSE-VOLUME oder KEEP-GENERATION angeben. Beachten Sie dazu auch Abschnitt „Kennungs-Überlaufschutz“ auf Seite 65 und Abschnitt „SYSLOG-Generationen erhalten“ auf Seite 66.

KEEP-GENERATION
alle Dateigenerationen bleiben erhalten, auch wenn die Anzahl in MAXIMUM überschritten wird.

CYCLIC-REPLACE und REUSE-VOLUME
bewirken, dass die älteste Dateigeneration der FGG vor dem Anlegen der neuen gelöscht wird.

Geben Sie hier CYCLIC-REPLACE oder REUSE-VOLUME an, dann sollten Sie den Wert für MAXIMUM (Anzahl der Dateigenerationen) nicht zu klein wählen. Nach dem Umschalten kann die "alte" Dateigeneration, die von *open*UTM nicht mehr beschrieben wird, noch über längere Zeit von einigen Tasks offengehalten werden (wenn diese sich sehr lange mit einem Anwender-Teilprogramm beschäftigen). Haben Sie nun *n* Dateigenerationen zugelassen und hält eine Task noch die Dateigeneration *i* offen, dann kann noch nicht auf die Dateigeneration *i+n* umgeschaltet werden. Das BS2000/OSD meldet einen DMS-Fehler für diese Dateigeneration. Die automatische Größenüberwachung wird suspendiert, bis der Administrator der Anwendung mit dem Kommando KDCSLOG die SYSLOG-Datei erfolgreich umschaltet.

Bei OVERFLOW-OPTION=KEEP-GENERATION ist jedoch auch in diesem Fall ein Umschalten möglich.

Mit dem Kommando legen Sie eine FGG an, deren Basis auf 0 liegt, d.h. außerhalb des gültigen Bereichs. *open*UTM legt alle Dateigenerationen selbst an. Beim ersten Start der Anwendung legt *open*UTM die Dateigeneration mit der Generationsnummer 1 an und benutzt diese als erste SYSLOG-Datei.

SYSLOG mit Link-Namen anlegen

Wollen Sie mit dem Link-Namen SYSLOG arbeiten, dann können Sie die SYSLOG mit demselben Kommando wie bei einer SYSLOG mit *filebase*.SLOG erzeugen, siehe oben. Für GROUP-NAME können Sie einen beliebigen Namen *fgg-name* angeben. Vor jedem Start der Anwendung müssen Sie dann der Dateigeneration, die *open*UTM als erste SYSLOG öffnen soll, den Link-Namen SYSLOG mit dem folgenden BS2000-Kommando zuweisen:

```
/SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=fgg-name(*gen)
```

gen ist die Generationsnummer der Dateigeneration, die *open*UTM nach dem Anwendungsstart als erste SYSLOG-Datei öffnen soll.

4.1.2.2 Dateigeneration anlegen

Vor dem Öffnen einer Dateigeneration überprüft *open*UTM, ob die entsprechende Dateigeneration schon existiert. Existiert die Dateigeneration nicht, dann legt *open*UTM die Dateigeneration an. Sie können die Dateigeneration jedoch auch selbst anlegen, z.B. um andere Werte für PRIMARY-ALLOCATION und SECONDARY-ALLOCATION zu definieren.

Anlegen der Dateigeneration durch *open*UTM

Abhängig davon, ob die FGG auf PUBLIC oder PRIVATE DISK angelegt wurde, setzt *open*UTM dazu intern eines der folgenden Kommandos ab:

- FGG auf PUBLIC DISK

```
/CREATE-FILE-GENERATION                          -
/   GENERATION-NAME = fgg-name(*gen)             -
/  ,SUPPORT = PUBLIC-DISK(                       -
/      SPACE = RELATIVE(                         -
/          PRIMARY-ALLOCATION = prim-alloc       -
/         ,SECONDARY-ALLOCATION = sec-alloc ) )
```

- FGG auf PRIVATE DISK

```
/CREATE-FILE-GENERATION                          -
/   GENERATION-NAME = fgg-name(*gen)             -
/  ,SUPPORT = PRIVATE-DISK(                      -
/      VOLUME = volume                           -
/     ,DEVICE-TYPE = device                      -
/     ,SPACE = RELATIVE(                         -
/          PRIMARY-ALLOCATION = prim-alloc       -
/         ,SECONDARY-ALLOCATION = sec-alloc ) )
```

Bedeutung der Parameter:

fgg-name Name der an *open*UTM übergebenen Dateigenerationsgruppe

gen Generation, die gerade zu öffnen ist

PUBLIC-DISK oder PRIVATE-DISK
gibt an, ob die Dateigeneration auf PUBLIC-DISK oder PRIVATE-DISK angelegt werden soll. Diese Information entnimmt *open*UTM dem Katalogeintrag der übergebenen SYSLOG-FGG.

VOLUME=volume, DEVICE=device
Datenträgerkennzeichen und Gerätetyp der Platte, auf der die FGG eingerichtet werden soll.

PRIMARY-ALLOCATION=prim-alloc, SECONDARY-ALLOCATION=sec-alloc
: Größe der Anfangszuweisung von Speicherplatz bzw. Größe der Speicherplatzerweiterung.

  Legt *open*UTM alle Generationen der FGG selbst an, dann setzt *open*UTM für beide Parameter 192 PAM-Seiten ein. Hat *open*UTM vor dem Anlegen der Dateigeneration bereits eine vorhandene (von Ihnen angelegte) Dateigeneration dieser FGG geöffnet, dann übernimmt *open*UTM deren Werte für Primary und Secondary Allocation beim Anlegen weiterer Dateigenerationen.

Dateigeneration selbst anlegen

Wollen Sie die Werte für Primary und Secondary Allocation für alle Dateigenerationen der SYSLOG-FGG selbst festlegen, dann müssen Sie also mindestens eine Dateigeneration mit den gewünschten Speicherplatz-Angaben selbst anlegen und diese Dateigeneration *open*UTM als erste SYSLOG-Datei übergeben. Alle weiteren Dateigenerationen werden von *open*UTM dann automatisch mit den angegebenen Werten für Primary und Secondary Allocation angelegt.

Die Dateigeneration können Sie mit dem folgenden BS2000-Kommando einrichten:

```
/CREATE-FILE-GENERATION                              –
/   GENERATION-NAME = filebase.SLOG(*1)              –
/  ,SUPPORT = PUBLIC-DISC(                           –
/      SPACE = RELATIVE(                             –
/         PRIMARY-ALLOCATION = prim-alloc            –
/        ,SECONDARY-ALLOCATION = sec-alloc  ) )
```

Liegt die FGG auf einer Privatplatte, dann müssen Sie die Dateigeneration auch auf Privatplatte anlegen.

Dateigeneration an openUTM übergeben

Übergeben Sie dann diese Dateigeneration als erste SYSLOG-Datei an *open*UTM. Dies kann entweder durch Verwenden des Link-Namens SYSLOG oder durch Festlegen der Basis der FGG (siehe Seite 60) geschehen.

Wenn Sie mit der zweiten Methode arbeiten, müssen Sie beachten, dass *open*UTM die Basis nicht verändert. Falls also die Basis der FGG im gültigen Bereich der FGG liegt und falls Sie die Basis vor dem nächsten Start der FGG nicht auf eine andere Generation legen, beginnt *open*UTM auch beim nächsten Anwendungsstart mit derselben (Basis-)Dateigeneration als erste SYSLOG-Datei. Die Informationen aus dem vorherigen Anwendungslauf gehen dann u.U. verloren. Darüber hinaus bleiben unabhängig davon, was Sie beim Einrichten der FGG für OVERFLOW-OPTION angegeben haben, nur die *n* jüngsten Dateigenerationen erhalten (*n*=MAXIMUM in CREATE-FILE-GROUP). Beachten Sie auch Abschnitt „SYSLOG-Generationen erhalten“ auf Seite 66.

Damit *open*UTM auch hier nach Anwendungsende und erneutem Start mit der nächsten Generation als erste SYSLOG anfängt, sollten Sie vor jedem Start folgende Schritte ausführen:

1. Die Basis auf die zuletzt beschriebene Generation der FGG (LAST-GEN) setzen

```
/MODIFY-FILE-GROUP-ATTRIBUTES                          -
/   GROUP-NAME = filebase.SLOG                         -
/  ,GENERATION-PARAMETER = GENERATION-PARAMETER(       -
/      BASE-NUMBER = RELATIVE-TO-LAST-GEN( 0 ) )
```

2. Die nächste Dateigeneration anlegen

```
/CREATE-FILE-GENERATION                                -
/   GENERATION-NAME = filebase.SLOG(+1)                -
/   ......
```

3. Die Basis auf die gerade angelegte Dateigeneration setzen

```
/MODIFY-FILE-GROUP-ATTRIBUTES                          -
/   GROUP-NAME = filebase.SLOG                         -
/  ,GENERATION-PARAMETER = GENERATION-PARAMETER(       -
/      BASE-NUMBER = RELATIVE-TO-LAST-GEN( 0 ) )
```

Diese Kommandofolge dürfen Sie nur einmal pro Anwendungsstart eingeben, nicht für jede gestartete UTM-Task.

4.1.2.3 Kennungs-Überlaufschutz

Sie kontrollieren die Belegung von Speicherplatz durch die SYSLOG-FGG wie folgt:

1. Schalten Sie die automatische Größenüberwachung ein, entweder bei der Generierung mit MAX...,SYSLOG-SIZE=*size*, oder über die Administration (z.B. mit dem Administrationskommando KDCSLOG SIZE= *size)*. Für *size* muss in beiden Fällen ein Wert > 0 angegeben werden.
2. Richten Sie die SYSLOG-FGG mit dem folgenden Kommando ein:

```
/CREATE-FILE-GROUP                                          -
/   GROUP-NAME = filebase.SLOG                              -
/  ,GENERATION-PARAMETER = GENERATION-PARAMETER(            -
/      MAXIMUM = n                                          -
/     ,OVERFLOW-OPTION = CYCLIC-REPLACE oder REUSE-VOLUME )
```

Die Dateigenerationen werden zyklisch überschrieben, so dass in der FGG maximal *n* Generationen katalogisiert sind. Zusätzlich kann es vorkommen, dass im BS2000/OSD intern noch eine weitere (ältere) Dateigeneration Speicherplatz belegt. Jede Generation hat durch die Größenüberwachung eine maximale Größe von *size* UTM-Seiten.

Damit ist der maximale Speicherverbrauch der SYSLOG-FGG kleiner oder gleich: (*n*+1) ∗ *size* ∗ (Größe einer UTM-Seite).

4.1.2.4 SYSLOG-Generationen erhalten

Wollen Sie alle Dateigenerationen der SYSLOG-FGG behalten, dann müssen Sie die SYSLOG-FGG wie folgt einrichten:

```
/CREATE-FILE-GROUP                                        -
/   GROUP-NAME = fgg-name                                 -
/  ,GENERATION-PARAMETER = GENERATION-PARAMETER(          -
/      MAXIMUM = n                                        -
/     ,OVERFLOW-OPTION = KEEP-GENERATION )
```

In diesem Fall steht die Basis auf 0, d.h. sie liegt außerhalb des gültigen Bereichs zwischen erster und letzter Dateigeneration. Das BS2000/OSD behält dann alle Generationen der FGG unabhängig von der Angabe im Parameter MAXIMUM (max. Anzahl der Generationen).



ACHTUNG!
Legen Sie die Basis der FGG jedoch in den gültigen Bereich, also zwischen erster und letzter Generation, dann gehen (ohne Warnung!!) alle Generationen, die außerhalb des gültigen Bereichs liegen verloren (Generationsnummern außerhalb von LAST-GEN - MAXIMUM und LAST-GEN).

Liegt die Basis innerhalb des gültigen Bereichs der FGG und wollen Sie trotzdem möglichst viele Generationen erhalten, dann müssen Sie *n* genügend groß (am besten 9999) wählen. In diesem Fall können Sie für den Operanden OVERFLOW-OPTION auch REUSE-VOLUME oder CYCLIC-REPLACE angeben.

4.1.2.5 Automatische Größenüberwachung

Die automatische Größenüberwachung können Sie nur für FGGs verwenden. Wenn Sie die SYSLOG-Datei als einfache Datei anlegen und die automatische Größenüberwachung generieren, dann bricht *open*UTM den Start der Anwendung mit Startfehlercode 58 ab.

Sie können die automatische Größenüberwachung auf zwei Arten einstellen:

- bei der Generierung mit der KDCDEF-Anweisung MAX ...,SYSLOG-SIZE=*size*
- im laufenden Betrieb der Anwendung mit dem Administrationskommando KDCSLOG [SWITCH,]SIZE=*size* oder an der Programmschnittstelle zur Administration mit dem Operationscode KC_SYSLOG und Subopcode KC_CHANGE_SIZE (siehe *open*UTM-Handbuch „Anwendungen administrieren")

In beiden Fällen müssen Sie für *size* einen Wert > 0 angeben.

Bei eingeschalteter Größenüberwachung überprüft *open*UTM vor jeder Meldungsausgabe in die SYSLOG-Datei, ob durch die Meldungsausgabe die vereinbarte Maximalgröße der Dateigeneration (*size* ∗ Größe einer UTM-Seite) überschritten würde. Ist dies der Fall, dann wird vor der Meldungsausgabe versucht, auf die nächste Dateigeneration zu schalten. Bei Erfolg gibt *open*UTM die Meldung K137 aus.

Tritt bei dem Versuch umzuschalten ein Fehler auf, dann arbeitet *open*UTM mit der alten Dateigeneration weiter, in die vor dem Umschaltversuch protokolliert wurde. *open*UTM schreibt die Meldung K139 auf SYSOUT und auf die Konsole des Administrators. Außerdem wird wie bei allen DMS-Fehlern zusätzlich die Meldung K043 ausgegeben. Sie enthält einen DMS-Fehlercode, dem Sie den Grund für den Umschaltfehler entnehmen können.

Damit *open*UTM nicht bei jeder folgenden Meldung mit dem Ziel SYSLOG erneut erfolglos versucht, auf die nächste Dateigeneration umzuschalten, wird die automatische Größenüberwachung nach einem solchen Umschaltfehler suspendiert.

Nachdem der Administrator den Grund für den Umschaltfehler gefunden und beseitigt hat, kann er die automatische Größenüberwachung wieder aktivieren, z.B. mit dem Kommando KDCSLOG SWITCH. Mit dem KDCSLOG SWITCH zwingt er *open*UTM, einen erneuten Umschaltversuch zu starten. Verläuft dieser Versuch ohne Fehler, so wird eine vorher suspendierte Größenüberwachung automatisch wieder aktiviert.

*open*UTM garantiert, dass nach dem Umschalten keine Meldung mehr in die alte Dateigeneration geschrieben wird. Das heißt jedoch nicht, dass Sie sofort frei über die alte Dateigeneration verfügen können. Erst müssen alle Tasks der Anwendung diese Dateigeneration schließen. Dies kann eventuell länger dauern, wenn Tasks sehr lange in Anwender-Teilprogrammen beschäftigt sind. Wird eine Dateigeneration von der letzten Task geschlossen, so gibt *open*UTM die Meldung K138 aus.

Welche Dateigenerationen geschlossen sind, können Sie der Ausgabe des Administrationskommandos KDCSLOG INFO entnehmen (LOWEST-OPEN-GEN).

*open*UTM verändert die vom Benutzer zu Anwendungsbeginn eingestellte FGG-Basis nicht. Dadurch gehen auch bei eingestellter FGG-Option OVERFLOW-OPTION= KEEP-GENERATION nicht ungewollt Dateigenerationen verloren.

4.1.3 Verhalten bei Schreibfehlern

Tritt bei dem Versuch, eine Meldung in die SYSLOG zu schreiben, ein Fehler auf, dann gibt *open*UTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen.

Das weitere Vorgehen von *open*UTM ist abhängig davon, ob die SYSLOG als einfache Datei oder als FGG geführt wird.

- Die SYSLOG wird als einfache Datei geführt

  Nach Ausgabe der Meldung K043 wird die Anwendung mit Grund SLOG09 abgebrochen.

- Die SYSLOG wird als FGG geführt

  *open*UTM versucht beim Auftreten eines Fehlers auf die nächste Dateigeneration zu schalten. *open*UTM schaltet auch um, wenn die Größenüberwachung ausgeschaltet bzw. nicht generiert ist. *open*UTM schaltet nicht um, wenn die Größenüberwachung auf Grund eines vorangegangenen Umschaltfehlers suspendiert ist.

  Schlägt der Umschaltversuch fehl, dann wird die Anwendung mit Grund SLOG09 abgebrochen.

  Kann *open*UTM auf die nächste Dateigeneration umschalten, dann versucht *open*UTM erneut die Meldung in die SYSLOG zu schreiben. Tritt dabei ein Fehler auf, wird die Anwendung mit SLOG09 abgebrochen. Tritt kein Fehler auf, läuft die Anwendung weiter, *open*UTM protokolliert in die neue SYSLOG-Dateigeneration.

4.1.4 SYSLOG-Datei auswerten

*open*UTM erstellt die SYSLOG-Datei im PAM-Format. Um sie lesbar ausdrucken oder mit einem Programm bearbeiten zu können, muss man die Datei aufbereiten. *open*UTM bietet dazu die Aufbereitungstools:

- KDCCSYSL

  wandelt die PAM-Datei in eine SAM-Datei um. KDCCSYSL wird in der Bibliothek SYSLNK.UTM.052.UTIL zur Verfügung gestellt.

- KDCPSYSL

  erzeugt aus der SAM-Datei eine weitere SAM-Datei mit den Meldungstexten. KDCPSYSL wird in der Bibliothek SYSLNK.UTM.052.UTIL zur Verfügung gestellt.

Um die Auswertung der SYSLOG-Datei zu erleichtern, werden mit *open*UTM die folgenden Beispielprozeduren ausgeliefert:

- SYSPRC.UTM.052(SYSLOG)

  Mit dieser Prozedur können Sie eine einfache SYSLOG-Datei bzw. eine einzelne Dateigeneration einer SYSLOG-FGG auswerten.

- SYSPRC.UTM.052(SLOG-FGG)

  Mit dieser Prozedur können Sie die einzelnen oder alle Dateigenerationen einer SYSLOG-FGG auswerten. Die Prozedur ruft in einer Schleife die beiden Tools KDCCSYSL und KDCPSYSL für die einzelnen Dateigenerationen auf. Die Auswertung kann auch bei laufender Anwendung erfolgen.

4.1.4.1 Das Tool KDCCSYSL - SYSLOG-Datei in SAM-Format umwandeln

Das Tool KDCCSYSL (Convert SYSLOG) wandelt die von *open*UTM als PAM-Datei erzeugte System-Protokolldatei in eine SAM-Datei um. Für jede Meldung erzeugt KDCCSYSL einen Satz variabler Länge. Der Aufbau wird durch Datenstrukturen KCMSG*x* beschrieben, die für alle Programmiersprachen bereitstehen, die *open*UTM unterstützt, beispielsweise KCMSGC für COBOL.

Aufruf von KDCCSYSL

```
/SET-FILE-LINK FILE-NAME=file_in,LINK-NAME=SLOGPAM   -
               [,SUPPORT=DISC(SHARED-UPDATE=YES)]
/CREATE-FILE FILE-NAME=file_out,
/SET-FILE-LINK FILE-NAME=file_out,LINK-NAME=SLOGSAM,ACCESS-METHOD=SAM
/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=SYSLNK.UTM.052.UTIL,-
                 ELEMENT=KDCCSYSL)
```

Bedeutung der Parameter:

file_in Name der SYSLOG-Datei

file_out Name, den die SAM-Datei bekommen soll

SHARED-UPDATE=YES ist im SET-FILE-LINK-Kommando anzugeben, wenn die SYSLOG-Datei aufbereitet werden soll, während die Anwendung läuft. Beachten Sie dabei aber, dass *open*UTM SYSLOG-Sätze zwischenspeichert, also nicht sofort in die PAM-Datei schreibt.

Soll die SYSLOG-Datei bei laufender Anwendung ausgewertet werden, dann empfiehlt es sich, vorher das Administrationskommando KDCSLOG WRITE einzugeben. Damit erreicht man, dass *open*UTM SYSLOG-Meldungen, die evtl. noch im virtuellen Speicher gepuffert sind, sofort auf die SYSLOG-Datei schreibt.

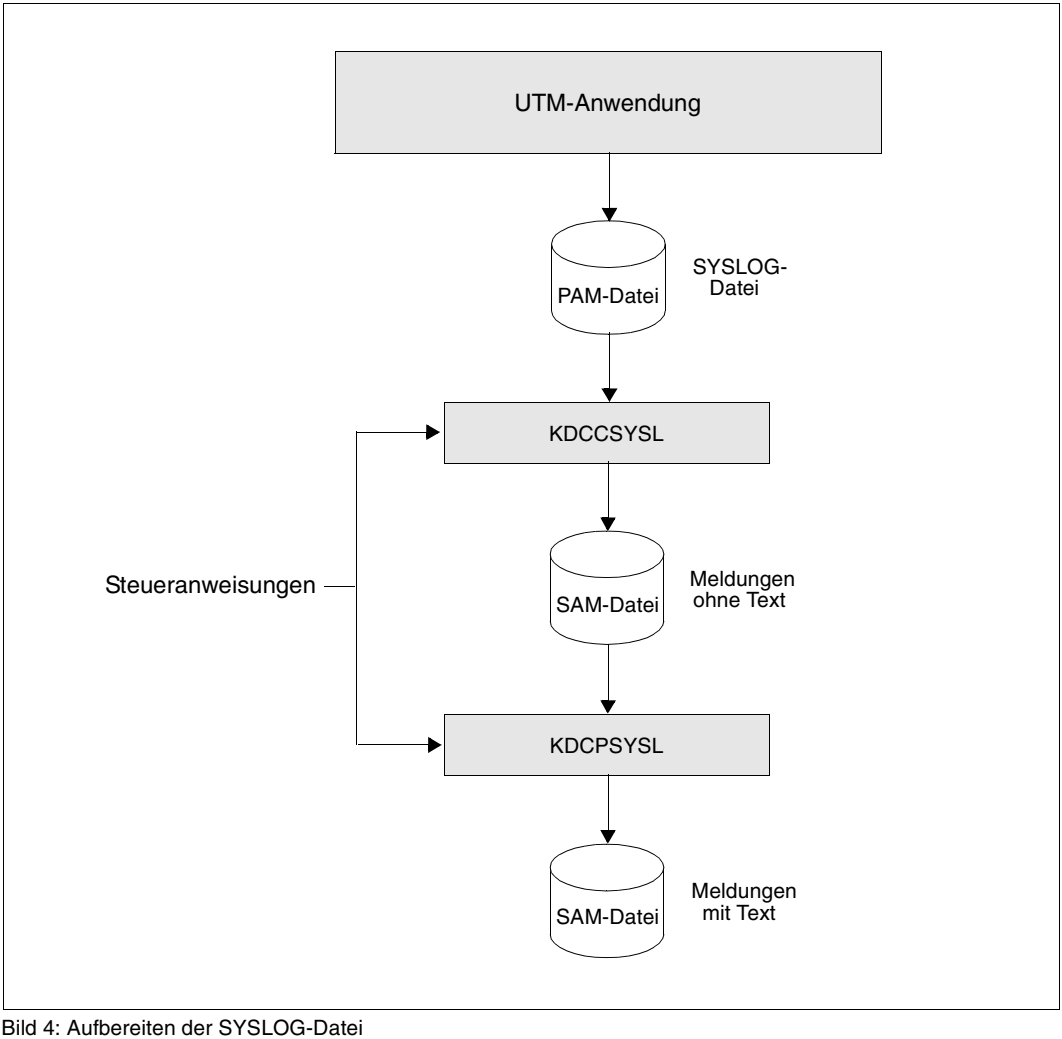


Bild 4: Aufbereiten der SYSLOG-Datei

4.1.4.2 Das Tool KDCPSYSL - Meldungstexte erzeugen

Das Aufbereitungstool KDCPSYSL liest die mit KDCCSYSL erzeugte SAM-Datei und erzeugt daraus eine SAM-Datei, die zu jeder Meldung die Meldungsnummer und den Meldungstext mit den aktuellen Textparametern angibt.

Datum und Uhrzeit werden am Anfang der Meldung ausgegeben. Die Inserts, die nicht schon im Meldungstext enthalten sind, werden am Ende hinzugefügt. Inserts, die hexadezimale Information liefern, werden dabei abdruckbar ausgegeben (z.B. BCAM-Infoword in der K065-Meldung; siehe *open*UTM-Handbuch „Meldungen, Test und Diagnose“).

Sie können die SYSLOG-Datei auch mit Meldungstexten aus einem eigenen Meldungsmodul aufbereiten lassen. Dazu geben Sie nach der Meldung

```
K558 Please enter name of user message module, a blank, or /EOF.
```

den Namen des eigenen Meldungsmoduls an. Wenn zuvor mit /SET-TASKLIB LIBRARY= eine benutzereigene Modulbibliothek zugewiesen wurde, sucht KDCPSYSL das angegebene Modul in dieser Bibliothek.

Wird auf die Meldung ein Leerzeichen (blank) oder "/EOF" eingegeben oder wird das Modul in der TASKLIB nicht gefunden, so bereitet KDCPSYSL die Meldungen mit dem Standard-Meldungsmodul von *open*UTM auf.

Die mit KDCPSYSL erzeugte Datei ist ebenfalls eine SAM-Datei, die pro Meldung eine oder mehrere Druckzeilen enthält. Diese Datei, die z.B. zur schnellen Übersicht dienen kann, können Sie mit dem PRINT-FILE-Kommando ausdrucken:

```
/PRINT-FILE FILE-NAME=file_out,LAYOUT-CONTROL=PARAMETERS(
            CONTROL-CHARACTERS=EBCDIC)
```

Aufruf von KDCPSYSL

```
/SET-FILE-LINK FILE-NAME=file_in,LINK-NAME=SLOGSAM
/CREATE-FILE FILE-NAME=file_out,
/SET-FILE-LINK FILE-NAME=file_out,LINK-NAME=SLOGEDIT,ACCESS-METHOD=SAM
[/SET-TASKLIB LIBRARY=omlname]
/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=SYSLNK.UTM.052.UTIL,-
ELEMENT=KDCPSYSL)
```

KDCPSYSL antwortet mit der Meldung:

```
K558 Please enter name of user message module, a blank, or /EOF
```

Geben Sie ein:

␣ oder
K2 und /EOF und /R oder
message_modul_name

Bedeutung der Parameter:

file_in Name der Ausgabedatei von KDCCSYSL (=Eingabedatei von KDCPSYSL)

file_out Name der Ausgabedatei von KDCPSYSL

omlname Name der Bibliothek (OML) mit dem eigenen Meldungsmodul

message_modul_name
Name des benutzereigenen Meldungsmoduls

4.1.4.3 Meldungen von KDCCSYSL und KDCPSYSL

Die Aufbereitungstools KDCCSYSL und KDCPSYSL schreiben ihre Meldungen nach SYSOUT.

Die Meldungen finden Sie im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.

4.2 Benutzer-Protokolldatei

In der Benutzer-Protokolldatei stehen die Sätze, die das Anwendungsprogramm mit LPUT-Aufrufen erzeugt hat. Die Benutzer-Protokolldatei muss als Dateigenerationsgruppe eingerichtet werden.

Die Benutzer-Protokoll-Sätze schreibt *open*UTM nicht sofort in die Protokolldatei, sondern speichert sie erst im Pagepool der KDCFILE. Sind im Pagepool so viele UTM-Seiten belegt, wie in MAX...,LPUTBUF=*number* generiert, kopiert *open*UTM die Sätze in die Benutzer-Protokolldatei. Das Kopieren erfolgt asynchron zu laufenden Transaktionen. Beim normalen Beenden der Anwendung kopiert *open*UTM die Sätze ebenfalls in die Benutzer-Protokolldatei.

Die Anzahl der bei LPUTBUF=*number* angegebenen UTM-Seiten ist bei der Generierung der Größe des Pagepools mit MAX...,PGPOOL=*number* zu berücksichtigen.

Mit MAX...,LPUTLTH=*length* beeinflussen Sie die Blocklänge der Benutzer-Protokolldatei. Sie wird von *open*UTM berechnet und kann größer sein als die Standardblockung 2KB.

*open*UTM kann LPUT-Sätze nur in die Benutzer-Protokolldatei kopieren, wenn diese eingerichtet ist und *open*UTM darauf zugreifen kann.

Bitte beachten Sie, dass die Benutzer-Protokolldatei nach einem KDCDEF-Lauf oder nach einem KDCUPD-Lauf von Anfang an überschrieben wird; ansonsten wird sie fortgeschrieben. Deshalb sollten Sie die Protokoll-Sätze vor einem KDCDEF- bzw. KDCUPD-Lauf auswerten.

4.2.1 Benutzer-Protokolldatei einrichten

Die Dateigenerationsgruppe für die Benutzer-Protokolldatei muss den Dateinamen *filebase*.USLA haben. *filebase* ist der in der MAX-Anweisung generierte Basisname der KDCFILE.

Die Dateigenerationsgruppe müssen Sie vor dem ersten Starten der Anwendung einrichten. Dazu sind folgende BS2000-Kommandos notwendig:

- CREATE-FILE-GROUP-Kommando zum Erstellen des Katalogeintrags. Sie müssen angeben:
  - den Namen der Dateigenerationsgruppe,
  - die maximal erlaubte Anzahl von Generationen,
  - die Vorgehensweise, wenn die maximale Anzahl Generationen erreicht ist (OVER-FLOW-OPTION=CYCLIC-REPLACE ist Standard)
  - evtl. die Zugriffsberechtigung.

- CREATE-FILE-GENERATION-Kommando zum Erstellen wenigstens einer Generation.
- MODIFY-FILE-GROUP-ATTRIBUTES-Kommando zum Festlegen der Bezugsgeneration für relative Nummerierung.

Beispiel: Dateigenerationsgruppe für eine Benutzer-Protokolldatei einrichten

```
CREATE-FILE-GROUP GROUP-NAME=filebase.USLA
                 ,GENERATION-PARAMETER=GENERATION-PARAMETER(
                  MAXIMUM=3,OVERFLOW-OPTION=REUSE-VOLUME)
CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(+1)
MODIFY-FILE-GROUP-ATTRIBUTES GROUP-NAME=filebase.USLA
                 ,GENERATION-PARAMETER=GENERATION-PARAMETER(
                  BASE-NUMBER=RELATIVE-TO-LAST-GEN(NUMBER=0))
```

Der Parameter OVERFLOW-OPTION=REUSE-VOLUME bewirkt, dass eine neue Generation auf demselben Datenträger eingerichtet wird wie die dafür gelöschte.

Falls die Benutzerprotokoll-Sätze (LPUT-Aufrufe) > 6 KB werden können, müssen Sie in den CREATE-FILE-GENERATION-Kommandos den SPACE-Operanden für die Primär- und Sekundärzuweisung angeben. Die Werte des SPACE-Operanden müssen so groß sein, dass mindestens ein LPUT-Satz in den durch die Sekundärzuweisung belegten Plattenbereich passt. Die Primärzuweisung muss mindestens doppelt so groß sein wie die Sekundärzuweisung.

Soll die Dateigenerationsgruppe auf einer Privatplatte eingerichtet werden, müssen Sie Folgendes zusätzlich beachten:

- Jede einzelne Dateigeneration der Gruppe müssen Sie vor dem Start der Anwendung mit einem CREATE-FILE-GENERATION-Kommando auf der Privatplatte einrichten.
- Sie müssen beim Einrichten der Dateigeneration OVERFLOW=REUSE-VOLUME angeben.
- Wird die Basis der Dateigenerationen nach dem Einrichten auf das letzte Element der Dateigenerationsgruppe gesetzt, beginnt *open*UTM mit der letzten Dateigeneration und schaltet danach auf die nächste Generation weiter.
  Wird als Basis die erste Generation angegeben, dann schaltet *open*UTM beim ersten Kommando KDCLOG (siehe Seite 75) auf die letzte Generation weiter.

Bei Dateigenerationen auf PUBLIC-DISK entfallen diese Einschränkungen.

Beispiel: Dateigenerationsgruppe auf Privatplatte einrichten

```
/CREATE-FILE-GROUP GROUP-NAME=filebase.USLA
                  ,GENERATION-PARAMETER=GENERATION-PARAMETER(
                   MAXIMUM=3,OVERFLOW-OPTION=REUSE-VOLUME
                  ,VOLUME=BOO4H,DEVICE-TYPE=D3465)
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(*1)
                  ,SUPPORT=PRIVATE-DISK(VOLUME=BOO4H,DEVICE-TYPE=D3465)
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(*2)
                  ,SUPPORT=PRIVATE-DISK(VOLUME=BOO4H,DEVICE-TYPE=D3465)
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(*3)
                  ,SUPPORT=PRIVATE-DISK(VOLUME=BOO4H,DEVICE-TYPE=D3465)
/MODIFY-FILE-GROUP-ATTRIBUTES GROUP-NAME=filebase.USLA
                  ,GENERATION-PARAMETER=GENERATION-PARAMETER(
                   BASE-NUMBER=RELATIVE-TO-LAST-GEN(NUMBER=0))
```

4.2.2 Doppelte Benutzer-Protokolldatei

Wird mit doppelten Benutzer-Protokolldateien gearbeitet (MAX...,USLOG=DOUBLE), so ist eine zweite Dateigenerationsgruppe mit dem Namen *filebase*.USLB auf gleiche Weise einzurichten wie die erste Dateigenerationsgruppe *filebase*.USLA. Die Dateigenerationsgruppe *filebase*.USLB muss auf demselben Katalog liegen wie die KDCFILE-Teile mit dem Suffix B.

4.2.3 Umschalten auf die nächste Dateigeneration

Zum Weiterschalten auf die nächste Dateigeneration stehen dem Administrator der Anwendung das Kommando KDCLOG und an der Programmschnittstelle KDCADMI der Opcode KC_USLOG zur Verfügung. Bei jedem Absetzen des KDCLOG-Kommandos bzw. eines entsprechenden KDCADMI-Aufrufs schaltet *open*UTM auf die jeweils folgende Dateigeneration weiter.

Beim ersten Start der UTM-Anwendung schreibt *open*UTM die Benutzer-Protokollsätze in die Generation der Gruppe, die beim Einrichten als Basis angegeben wurde. Bei jedem Weiterschalten durch den Administrator setzt *open*UTM ein MODIFY-FILE-GROUP-ATTRIBUTES-Kommando auf die nächste Dateigeneration ab und setzt die Basis auf die dann jüngste Dateigeneration.

Weiterschalten auf die nächste Generation können Sie auch wie folgt nach normaler Beendigung der UTM-Anwendung:

```
/CREATE-FILE-GENERATION GENERATION-NAME=filebase.USLA(+1)
/MODIFY-FILE-GROUP-ATTRIBUTES GROUP-NAME=filebase.USLA,
                              GENERATION-PARAMETER=GENERATION-PARAMETER(
                              BASE-NUMBER=RELATIVE-TO-LAST-GEN(NUMBER=0))
```

Diese Kommandos zum Weiterschalten auf die nächste Dateigeneration können Sie auch in den Start-Job Ihrer UTM-Anwendung übernehmen.

Will der UTM-Administrator z.B. mit KDCLOG auf eine andere Benutzer-Protokolldatei umschalten, dann muss die Dateigenerationsgruppe unter der gleichen BS2000-Benutzerkennung eingerichtet sein, unter der die UTM-Prozesse laufen.

4.2.4 Verhalten bei Schreibfehlern

Tritt beim Schreiben von LPUT-Sätzen in die Benutzer-Protokolldatei ein DMS-Fehler (**D**ata **M**anagement **S**ystem) auf, so gibt *open*UTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen. Gleichzeitig wird jeder weitere LPUT-Aufruf im Teilprogramm mit dem KDCS-Returncode 40Z (interner Returncode K903) abgewiesen.

Der Administrator der Anwendung kann dann die Benutzer-Protokolldatei bzw. ihre Generationen korrigieren, restaurieren oder neu einrichten.

Damit *open*UTM wieder LPUT-Sätze in die Benutzer-Protokolldatei schreibt, muss der Administrator der Anwendung nach der Fehlerbehebung das Administrationskommando KDCLOG oder einen KDCADMI-Aufruf mit Opcode KC_USLOG absetzen (siehe *open*UTM-Handbuch „Anwendungen administrieren”).

Die im Pagepool der KDCFILE gesicherten LPUT-Sätze werden jetzt in die Protokolldatei(en) geschrieben, die Dateigenerationsnummer wird erhöht.
Die Sperre für die LPUT-Aufrufe in den Teilprogrammen wird aufgehoben.

5 UTM-Anwendung starten

Eine UTM-Anwendung kann als ENTER-Task von der BS2000-Konsole oder von jedem BS2000-Terminal (Administratorterminal) gestartet werden.

Unter der dabei verwendeten BS2000-Benutzerkennung wird die CPU-Zeit usw. für die Anwendung abgerechnet.

Alle nötigen Dateien müssen entweder unter dieser Kennung katalogisiert sein oder mit SHARE=YES unter einer anderen Kennung auf demselben Host.

Benötigt werden die folgenden Dateien:

- Bibliothek mit einem Start-LLM oder mit einem ablauffähigen Großmodul
- KDCFILE
- Benutzer-Protokolldatei (USLOG, optional)
- System-Protokolldatei (SYSLOG)
- Benutzerdateien der Anwendung
- Modulbibliothek SYSLNK.UTM.052
- Bibliothek mit dem Verbindungsmodul für das Datenbanksystem (optional)
- Bibliothek mit dem Verbindungsmodul für das Formatierungssystem (optional)
- Bibliothek mit den Formaten (optional)

Die ENTER-Datei zum Starten der UTM-Anwendung enthält:

- CREATE-FILE- und SET-FILE-LINK-Kommando für System-Protokolldatei
- CREATE-FILE- und SET-FILE-LINK-Kommandos für Benutzerdateien (optional)
- START-PROGRAM-Kommando zum Aufruf des Anwendungsprogrammes
- Startparameter für UTM, FHS oder das Datenbanksystem

Gestartet wird die Anwendung durch den Aufruf des BS2000-Kommandos

```
/ENTER-JOB FROM-FILE=enterfile[,JOB-CLASS=job-class]
           [,RESOURCES=PARAMETERS(CPU-LIMIT=tttt)]
```

Empfehlungen für die Wahl der Parameter:

- Für den ENTER-Job einer UTM-Anwendung sollten Sie eine eigene Jobklasse einrichten, in der Sie die wichtigen Parameter des ENTER-Jobs einstellen.

  Die Jobklasse können Sie dann im ENTER-JOB-Kommando (JOB-CLASS=) oder beim LOGON-Kommando in der Enter-Datei zuweisen.

- Der Operand CPU-LIMIT (CPU-Zeitbeschränkung) sollte beim Aufruf der ENTER-Prozedur mitgegeben werden, wenn er nicht bereits im LOGON-Kommando in der Enter-Datei angegeben ist bzw. eine entsprechende Angabe bei der Definition der Jobklasse gemacht wird.

  Sie sollten den Wert so einstellen, dass die Jobs einer UTM-Anwendung keiner CPU-Zeitbeschränkung unterliegen. Aus diesem Grund sollten Sie CPU-LIMIT=NO (bzw. TIME=NTL) setzen bzw. die Jobklasse entsprechend definieren.

  Wird die CPU-Zeit beschränkt, d.h. CPU-LIMIT≠ NO gesetzt, und tritt für eine Task der Anwendung ein CPU-Time-Runout auf, dann kann dies zum abnormalen Ende der UTM-Anwendung führen!

  Bitte beachten Sie, dass für den Ablauf der Enter-Tasks ohne CPU-Zeitbegrenzung eventuell eine NTL-Berechtigung (**N**o **T**ime **L**imit) im JOIN-Eintrag für die betreffende Abrechnungsnummer notwendig ist.

  Eine Beschränkung des CPU-Verbrauchs von einzelnen Teilprogrammen einer UTM-Anwendung können Sie bei der KDCDEF-Generierung im Operanden TIME der TAC-Anweisung einstellen.

Beim Start der UTM-Anwendung werden alle Voraussetzungen für den Betrieb der Anwendung geschaffen, d.h. Bereiche und Tabellen eingerichtet, Dateien geöffnet, Verbindungen aufgebaut etc. Bei diesen Aktionen können Fehlersituationen auftreten, welche die Startroutine erkennt und die evtl. zum Abbruch des Starts der Anwendung oder einer Task führen. *open*UTM gibt dann die Meldung K078 oder K049 nach SYSOUT aus, die in einem Fehlercode die Ursache des Abbruchs anzeigt (siehe hierzu *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“).

In der Startprozedur sind keine Kommandos zur Konsistenzprüfung auf Dateien notwendig, die von *open*UTM geöffnet werden (REPAIR-DISK-FILES bzw. CHECK-FILE-CONSISTENCY). *open*UTM ruft beim Öffnen von Dateien, die nicht ordnungsgemäß geschlossen wurden, selbst diese Kommandos auf.

Das Anwendungsprogramm bleibt solange geladen, bis es entweder durch Eingriff des Administrators oder infolge eines Fehlers beendet wird. Wie *open*UTM das Anwendungsprogramm beendet, ist wichtig für den Aufbau der Startprozedur durch den Anwender.

Es gibt folgende Fälle:

- Nach einem Administrationskommando KDCSHUT NORMAL oder nach einem Abbruch der Anwendung (Term Application) beendet *open*UTM das Anwendungsprogramm mit TERM UNIT=STEP, d.h. in der ENTER-Prozedur werden alle Kommandos bis zum nächsten /SET-JOB-STEP bzw. bis /LOGOFF ignoriert.
- Beim Austausch des Anwendungsprogrammes mit KDCAPPL PROG=NEW oder nach einem schweren Programmfehler, der zum PEND ER führt, soll das Anwendungsprogramm neu geladen und gestartet werden. *open*UTM beendet das Programm dann mit TERM UNIT=PRGR, d.h. das nächste Kommando der ENTER-Prozedur wird interpretiert. Dort sollte ein Kommando /SKIP-COMMANDS stehen, das zum Kommando /START-PROGRAM für den Start des Anwendungsprogrammes zurückführt.

Der Anwender kann die Verarbeitung in der Startprozedur nach dem Kommando /SET-JOB-STEP davon abhängig machen, ob die UTM-Anwendung normal oder abnormal beendet wurde:

Bei einer abnormalen Beendigung der Anwendung (Term Application) erzeugt *open*UTM in der Task File Table (TFT) einen Eintrag mit LINK-NAME=KDCTRMAP. Man kann in der Startprozedur (mit /SHOW-FILE-LINK) abfragen, ob ein solcher Eintrag in der TFT vorhanden ist und abhängig davon den weiteren Ablauf steuern. Beim Schreiben eines UTM-Dumps wird dafür der LINK-Name KDCDUMP vergeben. In den Startprozeduren kann der Anwender z.B. sofort KDCDUMP dafür laufen lassen.
Eine andere Möglichkeit, *open*UTM nach einer abnormalen Beendigung neu zu starten, bietet der Jobvariablenanschluss (siehe Abschnitt „Nach abnormalem Anwendungsende erneut starten“ auf Seite 95).

Die Startprozedur ist in den folgenden Abschnitten beschrieben. Sie steht in einer Datei, die in diesem Kapitel mit *enterfile* bezeichnet wird. Ein Beispiel finden Sie auf Seite 92. Desweiteren kann eine UTM-Anwendung auch durch WinAdmin gestartet werden. Dazu muss neben WinAdmin auch das Produkt *open*FT installiert sein (im BS2000/OSD und auf dem WinAdmin-PC).

Mit der Startprozedur erzeugen Sie eine ENTER-Task. Sie können das ENTER-JOB-Kommando auch am Administratorterminal eingeben. Die benutzte Kennung ist dann im ENTER-JOB-Kommando oder im LOGON-Kommando der Startprozedur anzugeben.

Durch eine ENTER-Prozedur gestartete UTM-Tasks sind standardmäßig Batch-Tasks. Damit unterliegen sie den JOB-CLASS-Einschränkungen für Batch-Tasks. Im Normalfall will man jedoch UTM-Tasks sofort starten. Das lässt sich durch die Funktion "JOB EXPRESS" erreichen. Dazu muss man im JOIN-Eintrag der BS2000-Benutzerkennung, unter der die Tasks ablaufen sollen, EXP angeben.

Im BS2000/OSD gibt es neben den Batch- und Dialogtasks die Klasse der TP-Tasks. Sie behandelt das Betriebssystem bevorzugt. *open*UTM meldet die mit /ENTER-JOB gestarteten Tasks als TP-Tasks an. Diese Tasks werden aber nur dann bevorzugt behandelt, wenn unter der entsprechenden BS2000-Benutzerkennung bzw. Job-Klasse TP-Tasks erlaubt sind. Sind TP-Tasks verboten, laufen UTM-Tasks als Batch-Tasks.

Alle Tasks einer Anwendung müssen unter der gleichen BS2000-Benutzerkennung gestartet werden.



Eine UTM-Anwendung sollte nur zu Testzwecken im Dialog gestartet werden.

5.1 Startparameter der Anwendung

Beim Start des Anwendungsprogramms liest die Main Routine Startparameter von SYSDTA ein.

Mit den Startparametern werden aktuelle Ablaufparameter der Anwendung festgelegt. Dazu gehören z.B. die Anzahl der Tasks, mit der die Anwendung gestartet werden soll, oder Parameter für ein angeschlossenes Datenbank- und/oder Formatierungssystem.

Die Startparameter können in einer oder mehreren Zeilen angegeben werden. Ein Präfix bestimmt, für wen die Startparameter bestimmt sind:

- Startparameter mit dem Präfix „.UTM“ oder ohne Präfix werden von *open*UTM selbst interpretiert.
- Startparameter mit dem Präfix ".UDS", ".SESAM", ".LEASY", ".ORACLE", ".PRISMA", ".CIS" oder ".DB" leitet *open*UTM an das angeschlossene Datenbanksystem zur Auswertung weiter. Wenn die Anwendung z.B. mit DATABASE...,TYPE=UDS generiert wurde, ist als Präfix ".UDS" für die Parameter anzugeben, die an UDS weitergereicht werden sollen.
- Startparameter mit dem in der FORMSYS-Anweisung angegebenen Namen des Formatierungssystems als Präfix leitet *open*UTM an das Formatierungssystem FHS zur Auswertung weiter. Als Präfix ist ".FHS" anzugeben.

Wenn die Anwendung mit Nutzung der Lade-Modul-Schnittstelle generiert ist (KDCDEF-Anweisung LOAD-MODULE), dann sind die Startparameter in zwei Blöcken anzugeben:

- 1. Block: Startparameter für *open*UTM
- 2. Block: Startparameter für Datenbank- und Formatierungssystem

Innerhalb eines Blocks ist die Reihenfolge der Startparameter beliebig. Jeder Block - auch ein leerer Block - wird mit einem END-Kommando abgeschlossen. Wenn beispielsweise kein Datenbank- und Formatierungssystem generiert ist, folgen zwei END-Kommandos aufeinander.

UTM-Startparameter im zweiten Block werden ignoriert. Das gleiche gilt für Startparameter des Datenbank- und Formatierungssystems im ersten Block.

5.1.1 Startparameter für *open*UTM

Die Syntax der UTM-Startparameter ist im Folgenden dargestellt.

```
[.UTM] START   FILEBASE=filebase
               [ ,ASYNTASKS=number ]
               [ ,BTRACE={ ON | OFF } ]
               [ ,CATID=(catalog-A,catalog-B) ]
               [ ,DBKEY=db-key ]
               [ ,DUMP-CONTENT={ STANDARD | EXTENDED } ]
               [ ,DUMP-MESSAGE=event-code]
               [ ,DUMP-USERID={ STANDARD | SYSUSER } ]
               [ ,OTRACE={ ON | (SPI, INT, OSS, SERV, PROT) | OFF } ]
               [ ,PASSWORD=connection-password ]
               [ ,ROOTNAME=rootname ]
               [ ,STARTNAME={ enterfile |
                              'enterfile[,enteroperand][...]' } ]
               [ ,STXIT={ ON | OFF } ]
               [ ,TABLIB=libname ]
               [ ,TASKS=number ]
               [ ,TASKS-IN-PGWT=number ]
               [ ,TESTMODE={ ON | OFF | FILE } ]
               [ ,UTM-MSG-DATE={ YES | NO } ]

[.UTM] END
```

In obiger Syntax werden die Parameter in einer Zeile ohne Zeilenvorschub angegeben und jeweils durch ein Komma getrennt.

Sie können die Parameter beim START-Kommando jedoch auch auf mehrere Zeilen verteilen. In diesem Fall müssen Sie in jeder Zeile das Kommando START vor den Parameter stellen.

Syntax-Prüfung beim Start der Anwendung

- Erkennt *open*UTM bei der Überprüfung der Startparameter einen Syntaxfehler, dann gibt *open*UTM die Meldung K038 aus, setzt für den betroffenen Startparameter den Standardwert, sofern vorhanden, und startet die Anwendung.
- Bei einem Syntaxfehler im Parameter FILEBASE kann die Anwendung **nicht** gestartet werden, da kein Standardwert für diesen Parameter existiert.
- Bei einem Syntaxfehler im Parameter PASSWORD wird der Standardwert (= kein Passwort) gesetzt. D.h die Verbindung zur UTM-Anwendung kann dann ohne Passwortangabe aufgebaut werden!

Bedeutung der Kommandos

START Mit diesem Kommando gibt man die für den Lauf einer UTM-Anwendung erforderlichen UTM-Startparameter an. Die Anwendung wird nach der Eingabe aller Startparameter sofort gestartet.

END Dieses Kommando schließt die Eingabe der Startparameter ab (*open*UTM, Datenbank- und Formatierungssystem).

Die Startparameter müssen in zwei Blöcken angegeben werden:
– zuerst die *open*UTM-Startparameter
– dann die Startparameter für die angeschlossenen Datenbank- und Formatierungssysteme.

Jeder Block wird mit dem END-Kommando abgeschlossen.

Bedeutung der Operanden

FILEBASE=filebase

Basisname für die KDCFILE und die Benutzer-Protokolldatei, wie er bei der KDCDEF-Generierung mit MAX...,KDCFILE= angegeben wurde. Bei Angabe eines ungültigen Namens wird der Start der Anwendung abgebrochen.

ASYNTASKS=number

Anzahl der Tasks, die max. für asynchrone Vorgänge arbeiten sollen.

Standard: in MAX...,ASYNTASKS=*number* festgelegte Anzahl
Minimalwert: 0
Maximalwert: in MAX...,ASYNTASKS=*number* festgelegte Anzahl

BTRACE= Ein-/Ausschalten der BCAM-Tracefunktion.

ON Beim Start der Anwendung wird die BCAM-Tracefunktion eingeschaltet. Es werden alle verbindungsspezifischen Ereignisse in der BCAM-Tracedatei aufgezeichnet. Sehen Sie dazu auch das *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.

OFF Beim Start der Anwendung bleibt die BCAM-Tracefunktion ausgeschaltet.

Standard: OFF

CATID=(catalog-A,catalog-B)

Mit diesem Parameter können Sie die Katalogkennungen angeben, denen die Teile der KDCFILE zugeordnet werden. Wenn Sie mit CATID arbeiten, müssen Sie im Parameter FILEBASE den Basisnamen ohne CATID angeben. Bei doppelter Dateiführung können Sie die Dateien mit Suffix A *catalog-A* und die Dateien mit Suffix B *catalog-B* zuordnen. Arbeiten Sie mit doppelter Dateiführung und geben nur *catalog-A* an, werden beide Dateien derselben CATID zugeordnet.

DBKEY=db-key

Bis zu acht Zeichen langer Schlüssel für den Zugriff zur Datenbank. *db-key* wird bei Aufrufen an das Datenbanksystem verwendet, wenn für den Transaktionscode des aufrufenden Teilprogramms kein DBKEY definiert ist.

Standard: C'UTM␣␣␣␣␣'

DUMP-CONTENT=

gibt an, ob *open*UTM die taskübergreifenden Speicherbereiche in allen Dumps einer Dump-Dateigeneration, d.h. für alle Tasks, abzieht oder nur im Dump der Task, die den Anwendungsabbruch verursacht hat.

STANDAR**D**

Wenn *open*UTM eine Dump-Dateigeneration erzeugt, dann sind taskübergreifende Speicherbereiche nur im Dump der ersten Task (Verursacher) enthalten. Für die Diagnose ist das normalerweise ausreichend.

Standard: STANDARD

EXTENDED

Die taskübergreifenden Speicherbereiche sind in allen Dumps einer DUMP-Dateigeneration enthalten.



Diesen Wert sollten Sie nur auf besondere Anforderung des Service einstellen.

DUMP-MESSAGE= *event-code*

event-code gibt das Ereignis an, bei dem UTM bei eingeschaltetem Testmodus einen UTM-Dump mit Kennzeichen MSGDMP erzeugt. Ein Dump wird nur von der Task erstellt, in der das Ereignis eingetreten ist; die Anwendung wird dabei nicht beendet.

Der Wert von DUMP-MESSAGE kann per Administration zurückgesetzt werden, z.B. durch WinAdmin oder das Kommando KDCDIAG DUMP-MESSAGE=K000.

Sie können für *event-code* Folgendes angeben:

– *event-code*=K*nnn* (K-Meldung)

  Der UTM-Dump wird erzeugt, wenn die Meldung K*nnn* auftritt.
  Bei den Meldungsnummern K023, K043, K061, K062 wird nur einmal ein Dump erzeugt, danach wird *event-code* automatisch zurückgesetzt.
  Bei allen anderen Meldungen wird solange bei jedem Auftreten der Meldungsnummer ein Dump erzeugt, bis der Wert per Administration (s.o.) zurückgesetzt wird.

– *event-code*=CC*nn* (Kompatibler KDCS-Returncode)

  Der UTM-Dump wird erzeugt, wenn in einem Teilprogramm ein KDCS-Returncode der Form *nn*Z auftritt, z.B 14Z. Danach wird der Wert zurückgesetzt.

– *event-code*=K*xyz* (Inkompatibler KDCS-Returncode)

  Der UTM-Dump wird erzeugt, wenn in einem Teilprogramm der inkompatible KDCS-Returncode K*xyz* auftritt, z.B KD10. Danach wird der Wert zurückgesetzt.

– *event-code*=S*xyy* (SIGN-Returncode)

  Der UTM-Dump wird erzeugt, wenn beim SIGN-Aufruf ein Returncode der Form *xyy* auftritt. Hierbei sind *x* und *yy* die Werte in KCRSIGN1 bzw. KCRSIGN2, wobei KCRSIGN1 den Wert U, I, A oder R haben muss. SU13 bedeutet z.B. „neues Passwort zu kurz“. Der Dump wird unabhängig davon erzeugt, ob in der Anwendung ein Anmeldevorgang generiert ist oder nicht. Anschließend wird der Wert zurückgesetzt.

Hinweis:
Bei allen KDCS-Returncodes ≥70Z und den zugehörigen inkompatiblen KDCS-Returncodes, bei denen kein PENDER-Dump geschrieben wird (z.B. 70Z/K316), wird auch kein DUMP mit Kennzeichen MSGDMP erzeugt.

DUMP-USERID=
gibt an, unter welcher Kennung die UTM-Dumps abgelegt werden sollen.

STANDAR**D**
Die UTM-Dumps werden unter der Kennung abgelegt, unter der die UTM-Task gestartet wurde.

Standard: STANDARD

SYSUSER
Die UTM-Dumps werden unter der Kennung $SYSUSER abgelegt.

OTRACE= Ein-/Ausschalten der Tracefunktion von OSS.
Der OSS-Trace wird zur Diagnose bei Problemen mit OSI-TP-Verbindungen der Anwendung benötigt. Sehen Sie dazu auch das *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“ und das Handbuch zu OSS.

ON Schaltet die OSS-Tracefunktion ein.
Es werden die Trace-Records der Typen SPI, INT, OSS, SERV und PROT protokolliert. Beim Einschalten der OSS-Tracefunktion erzeugt jede Task der Anwendung seine eigene Trace-Datei.

(SPI, INT, OSS, SERV, PROT)
Schaltet die OSS-Tracefunktion ein. Es werden die Trace-Records des angegebenen Typs protokolliert. Die Reihenfolge in der die Trace-Records angegeben werden ist beliebig.

SPI
Das XAP-TP System Programming Interface wird protokolliert.

INT
Der interne Ablauf im XAP-TP-Baustein wird protokolliert.

OSS
Die OSS-Aufrufe werden protokolliert.

SERV
Die OSS-internen Trace Records vom Typ O_TR_SERV werden protokolliert.

PROT
Die OSS-internen Trace Records vom Typ O_TR_PROT werden protokolliert.

OFF Schaltet die OSS-Tracefunktion aus.

PASSWORD=connection-password
ein 4 Zeichen langes Kennwort, das der Benutzer zum Aufbau der logischen Verbindung mit der Anwendung angeben muss (siehe Abschnitt „Standard-Anmeldeverfahren für Terminals“ auf Seite 112).

Standard: kein Kennwort (X’0000 0000’)

ROOTNAME=rootname
Der Parameter ist Pflicht, wenn der ROOT-Tabellenmodul nachgeladen werden soll. *rootname* ist der PLAM-Elementname des ROOT-Tabellenmoduls in der bei TABLIB= angegebenen Bibliothek.

STARTNAME=

enterfile Name der im /ENTER-Kommando angegebenen Datei. Die Angabe ist nur bei Mehrtask-Betrieb (TASKS > 1) notwendig. Die ENTER-Datei muss unter der Benutzerkennung katalogisiert sein, unter der die Anwendung gestartet wurde. *enterfile* darf maximal 54 byte lang sein.

enteroperand
Operand des BS2000-Kommandos /ENTER. *open*UTM startet die Folgetasks und setzt dabei die angegebenen Operanden ein. Schreibweise und Wirkung der angegebenen Operanden entsprechen den Operanden des /ENTER-Kommandos. Die Operanden müssen im ISP-Format eingegeben werden.

Die Zeichenfolge zwischen den Hochkommata
(*'enterfile, enteroperand,...'*) darf maximal 200 Zeichen lang sein.

STXIT= STXIT-Routinen aktivieren.

ON Die UTM-STXIT ist eingeschaltet.

Standard: ON

OFF Die UTM-STXIT wird ausgeschaltet. Dies ist nur möglich für UTM-Anwendungen, die im Dialog gestartet werden.

TABLIB=libname

Der Parameter ist Pflicht, wenn der ROOT-Tabellenmodul nachgeladen werden soll. Die Bibliothek *libname* muss dann das Objekt des ROOT-Tabellenmoduls enthalten.

TASKS=number

Anzahl der BS2000-Tasks, die für die Anwendung gestartet werden sollen.

Standard: in MAX...,TASKS=*number* festgelegte Anzahl
Minimalwert: 1
Maximalwert: in MAX...,TASKS=*number* festgelegte Anzahl

Soll die Anwendung mit mehreren Tasks arbeiten, ist auch der Startparameter STARTNAME= anzugeben.

TASKS-IN-PGWT=number

Maximale Anzahl der Tasks, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen wie z.B. der KDCS-Aufruf PGWT ablaufen dürfen (Operand PGWT= in den KDCDEF-Anweisungen TAC und TACCLASS).

Standard: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.
Minimalwert: 1 falls MAX...,TASKS-IN-PGWT > 0 generiert wurde, sonst 0.
Maximalwert: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.

TESTMODE=

Testmodus einschalten.
Sehen Sie hierzu auch *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“, Fehlerdiagnose.

ON Beim Start der Anwendung soll der Testmodus eingeschaltet werden. Im Testmodus werden zusätzliche UTM-interne Plausibilitätsprüfungen bei internen Prozeduraufrufen durchgeführt und interne Trace-Informationen sowohl im KTA als auch im XAP-TP-Baustein bei OSI-TP-Anwendungen aufgezeichnet. Der Testmodus sollte nur zur Diagnose von UTM-Fehlern auf Empfehlung des Systemberaters eingeschaltet werden.

OFF Beim Start der Anwendung soll der Testmodus ausgeschaltet bleiben. In bestimmten Fällen werden bei TESTMODE=OFF Dumps nach PEND ER unterdrückt.

Standard: OFF

FILE Beim Start der Anwendung wird der Testmodus eingeschaltet. Zusätzlich werden bei jedem Überlauf der KTA-Trace-Area die Diagnosedaten auf Datei geschrieben, um einen evtl. Diagnosedatenverlust zu vermeiden. Der Dateiname setzt sich zusammen aus dem Basisnamen *filebase* und der TSN der jeweiligen Task, d.h. es wird bei einer UTM-Produktivanwendung pro Task folgende Datei angelegt:

filebase.KTATRC.*tsn*

UTM-MSG-DATE=

gibt an, ob UTM-Meldungen mit oder ohne Datum/Uhrzeit auf SYSOUT/SYSLST ausgegeben werden.

YES Alle UTM-Meldungen auf SYSOUT/SYSLST werden mit dem Datum/Uhrzeit-Präfix ausgezeichnet.

Standard: YES

NO Die UTM-Meldungen erhalten kein Datum/Uhrzeit-Präfix.

5.1.2 Startparameter für das Datenbanksystem

Die Schreibweise und die Bedeutung dieser Parameter ist in den Benutzerhandbüchern der entsprechenden DB-Systeme beschrieben. Zusätzliche Informationen finden Sie auch in Abschnitt „UTM-Datenbank-Anwendung starten und beenden“ auf Seite 105.

5.1.3 Startparameter für das Formatierungssystem

Die Schreibweise und die Bedeutung dieser Startparameter sind im Benutzerhandbuch des Formatierungssystems (FHS) beschrieben. *open*UTM akzeptiert Startparameter für das Formatierungssystem nur dann, wenn dieses in der KDCDEF-Anweisung FORMSYS bei der Generierung angegeben wurde.

5.2 Starten der Anwendung

Das Anwendungsprogramm wird durch das Kommando /START-PROGRAM gestartet. Dabei gilt Folgendes:

- Falls shareable Teile der Anwendung in den Systemspeicher geladen werden sollen, so muss dies noch vor dem Start des Anwendungsprogramms durch den Administrator veranlasst werden.
- Das /START-PROGRAM-Kommando bewirkt, dass der statisch gebundene Teil des Anwendungsprogramms in den Arbeitsspeicher geladen wird. Alle Lademodule des Anwendungsprogramms, die mit LOAD-MODE=STARTUP oder LOAD-MODE=(POOL,...) ... bzw. LOAD=STARTUP oder LOAD=(POOL,...) generiert wurden, werden beim Start der Anwendung als eigenständige Einheiten nachgeladen.
- Können einzelne Lademodule nicht geladen werden, so wird i.a. der Start der Task fortgesetzt. Soll im Betrieb der Anwendung ein Programm angesprungen werden, das nicht geladen werden konnte, so führt dies zum Aufruf des Event-Services BADTACS oder zu einem PEND ER. Wenn die Event-Services MSGTAC, SIGNON oder die Event-Exits START, SHUT, INPUT und FORMAT oder das Administrations-Teilprogramm oder AREAs nicht geladen werden können, wird der Start der Task mit einer Fehlermeldung abgebrochen.
- Beim Starten einer zusätzlichen Task für die Anwendung sowie nach einem PEND ER prüft *open*UTM, ob das in dieser Task geladene Anwendungsprogramm mit dem in den zuvor gestarteten Tasks übereinstimmt. *open*UTM überprüft die Generierungsinformationen. Bei fehlender Übereinstimmung bricht *open*UTM den Start der Folge-Task mit einer Fehlermeldung ab.

Startkommandos für ein mit BINDER gebundenes Anwendungsprogramm

Das Anwendungsprogramm wurde mit dem BINDER des BLS gebunden. Die Anwendung wurde für den Einsatz des BLS beim Nachladen und Austauschen von Anwendungsteilen generiert. In diesem Fall wird das Anwendungsprogramm gestartet mit:

```
/START-PROGRAM FROM-FILE=*MODULE                                    -
               (  LIBRARY=llm-plamlib                               -
                , ELEMENT=llm-rootname                              -
                , PROGRAM-MODE=ANY/24)                              -
                , RUN-MODE=ADVANCED                                 -
                           (UNRESOLVED-EXTRNS=DELAY                 -
                           ,LOAD-INFORMATION=REFERENCES)            -
               )
```

Das mit dem BINDER gebundene Anwendungsprogramm *llm-rootname* muss als Element vom Typ L in einer Programmbibliothek *llm-plamlib* bereitgestellt werden.

Wenn Sie ein Lademodul mit ALTERNATE-LIBRARIES=YES generiert haben, dann müssen Sie vor dem Start des Anwendungsprogramms der Bibliothek des betreffenden Laufzeitsystems einen Linknamen (BLSLIB*nn* mit 00≤*nn*≤99) zuweisen. Sie können sich mit dem BS2000-Kommando

```
/MODIFY-DBL-PARAMETERS SCOPE=*ALL(PROGRAM-MAP=...)
```

eine DBL-Liste über die dynamischen Ladevorgänge während des Betriebs der Anwendung ausgeben lassen. Da dies aber den Startvorgang bzw. den Programmaustausch verzögert, sollten Sie von dieser Möglichkeit nur im Test- oder Fehlerfall Gebrauch machen.

Wenn Teile des Anwendungsprogramms zu einem späteren Zeitpunkt nachgeladen werden sollen, müssen Sie beim Starten die folgenden Operanden angeben:

```
RUN-MODE=*ADVANCED (UNRESOLVED-EXTRNS = *DELAY,        -
                    LOAD-INFORMATION  = *REFERENCES)
```

Wenn Sie die Autolink-Funktion während des Starts nicht benötigen und Sie im Start-LLM unbefriedigte Externverweise haben, sollten Sie die Funktion mit AUTOLINK=*NO unterdrücken. Auch wenn das Start-LLM beim Start keine offene Externverweise auf den Shared Code besitzt, sollte das Suchen des DBL mit dem Startoperanden SHARE-SCOPE=*NONE unterbunden werden.

Beim Start der Anwendung lädt *open*UTM zunächst alle Lademodule, die shareable in Common Memory Pools (LOAD-MODE=POOL) gehalten werden sollen. Dabei werden zuerst die Common Memory Pools geladen, die mit SCOPE=GLOBAL generiert wurden und danach diejenigen mit SCOPE=GROUP. Anschließend lädt *open*UTM alle Lademodule, die mit LOAD-MODE=STARTUP generiert wurden, und zwar in der Reihenfolge, in der Sie die LOAD-MODULE-Anweisungen bei der Generierung angegeben haben.

Lademodule, die mit LOAD-MODE=ONCALL generiert wurden, werden erst beim Aufruf eines Teilprogramms dieses Lademoduls geladen.

i Zwei UTM-Anwendungen sollten möglichst unter unterschiedlichen BS2000-Benutzerkennungen gestartet werden, um Fehler zu vermeiden, die wegen gleicher Modulnamen in shareable Teilen auftreten können. Module, die von mehreren Anwendungen benutzt werden, sollten daher in anwendungsglobale Common Memory Pools oder in nichtprivilegierte Subsysteme geladen werden.

5.3 Kaltstart und Warmstart

Man versteht bei *open*UTM unter:

- Kaltstart:
  Starten nach normaler Beendigung der UTM-Anwendung oder nach Neugenerierung.
- Warmstart:
  Starten nach abnormaler Beendigung der UTM-Anwendung.

Kaltstart bei *open*UTM

Vor dem ersten Start einer Anwendung haben Sie die KDCFILE mit dem Generierungstool KDCDEF eingerichtet. Wenn die UTM-Anwendung normal beendet wurde, greift *open*UTM bei einem erneuten Start der Anwendung wieder auf dieselbe KDCFILE zu und führt einen Kaltstart durch. Nach erfolgreichem Start meldet *open*UTM:

```
K051 Kaltstart für Anwendung appliname mit UTM V05.2A00 erfolgreich
```

Warmstart bei *open*UTM

Bei einem Wiederanlauf bringt *open*UTM die KDCFILE in einen konsistenten Zustand. Nach erfolgreichem Start meldet *open*UTM:

```
K050 Warmstart für Anwendung appliname mit UTM V05.2A00 erfolgreich.
```

Dabei ist zu beachten, dass sich UTM-S und UTM-F im Umfang der Wiederanlauf-Funktionen unterscheiden. Sehen Sie dazu auch im *open*UTM-Handbuch „Konzepte und Funktionen“; Fehlertoleranz, Wiederanlauf.

5.4 Beispiel einer UTM-Startprozedur

Im folgenden Beispiel finden Sie eine Startprozedur, die eine Anwendung für den Produktivbetrieb startet.

```
/SET-PROCEDURE-OPTIONS                                    -
/       ,DATA-ESC             = STD                       -
/       ,IMPLICIT-DECLARATION = NO
/
/ " ------------------------------------------------------------------- "
/ "                                                                     "
/ " PROCEDURE: START-BLS-APPLICATION                                    "
/ "                                                                     "
/ "                                                                     "
/ " Purpose:                                                            "
/ "   This procedure creates an enter procedure to start an             "
/ "    UTM BLS application.                                             "
/ "                                                                     "
/ " Environment:                                                        "
/ "    SDF-P  V1.0                                                      "
/ "    JV     V11.0 (Only $SYSJV.TSN to determine the TSN of            "
/ "                      the ENTER-JOB )                                "
/ "                                                                     "
/ "  ASYNTASKS:                                                         "
/ "                                                                     "
/ " Parameter:                                                          "
/ "                                                                     "
/ "  ASYNTASKS:                                                         "
/ "       Maximal number of asynchronus tasks                           "
/ "  FILEBASE:                                                          "
/ "       Base name of KDCFILE and SYSLOG                               "
/ "  MAPLIB:                                                            "
/ "       Format library                                                "
/ "  PASSWORD:                                                          "
/ "       Connection password for application                           "
/ "  ROOTNAME = *NONE/element-name                                      "
/ "       *NONE: Dynamic root is not wanted                             "
/ "       element-name:                                                 "
/ "              If dynamic root is wanted this is the root table       "
/ "              module in the library TABLIB.                          "
/ "  START:                                                             "
/ "       YES/NO Starting the application after creation               "
/ "  START-LLM-LIB:                                                     "
/ "       Name of the library with the START LLM                        "
/ "  START-LLM-NAME:                                                    "
/ "       Name of the START LLM                                         "
/ "  STARTNAME:                                                         "
/ "       Name of the ENTER file                                        "
```

```
/ "  TABLIB = *NONE/library-name                                          "
/ "       *NONE: Dynamic root is not wanted                               "
/ "       library-name:                                                   "
/ "              If dynamic root is wanted this is the library            "
/ "              of the root table module (see parameter                  "
/ "              ROOTNAME).                                               "
/ "  TASKS:                                                              "
/ "       Maximal number of tasks of the application                      "
/ "  TESTMODE:                                                           "
/ "       OFF/ON  long UTM trace.                                         "
/ "                                                                       "
/ "                                                                       "
/ " ------------------------------------------------------------------- "
/BEGIN-PARAMETER-DECLARATION
/  DECL-PAR  ASYNTASKS       (                 , INTEGER )
/  DECL-PAR  FILEBASE        (                 , STRING  )
/  DECL-PAR  MAPLIB          (                 , STRING  )
/  DECL-PAR  PASSWORD        ( 'X''00000000''' , STRING  )
/  DECL-PAR  ROOTNAME        ( '*NONE'         , STRING  )
/  DECL-PAR  START           (                 , BOOLEAN )
/  DECL-PAR  START-LLM-LIB   (                 , STRING  )
/  DECL-PAR  START-LLM-NAME  (                 , STRING  )
/  DECL-PAR  STARTNAME       (                 , STRING  )
/  DECL-PAR  TABLIB          ( '*NONE'         , STRING  )
/  DECL-PAR  TASKS           (                 , INTEGER )
/  DECL-PAR  TESTMODE        ( 'ON'            , STRING  )
/END-PARAMETER-DECLARATION
/
/
/IF IS-CATALOGED-FILE( STARTNAME )
/   DELETE-FILE FILE-NAME=&STARTNAME
/   END-IF
/IF IS-CATALOGED-FILE( 'SYSOUT.&STARTNAME' )
/   DELETE-FILE FILE-NAME=SYSOUT.&STARTNAME
/   END-IF
/
/ EDT: BEGIN-BLOCK DATA-INSERTION = YES
/
/MODIFY-JOB-SWITCHES ON=(4,5)
/MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=NO-CONTROL
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/START-PROGRAM FROM-FILE=$EDT
*/        SET-LOGON-PARAMETERS , JOB-NAME = &FILEBASE
*/        SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=&FILEBASE..SYSLOG
*/        MODIFY-TEST-OPTION DUMP=YES
*/        ASSIGN-SYSDTA TO-FILE=*SYSCMD
*/        ASSIGN-SYSOUT TO-FILE=SYSOUT.&&($SYSJV.TSN).&FILEBASE
/
/ " ------------------------------------------------------------------- "
/ "  Here follows the section with the SET-FILE-LINK commands if        "
/ "   at least one LOAD-MODULE is defined with                          "
/ "   ALTERNATE-LIBRARIES = YES.                                        "
/ "  If you need RFA connections, or an AID connection (AID-FE)         "
/ "   for testing an UTM production application or other                "
/ "   connections please insert here the commands.                      "
/ " ------------------------------------------------------------------- "
/
/
```

```
*/        SKIP-COMMANDS TO-LABEL=EXEC
*/.REPEAT SHOW-FILE-LINK LINK-NAME=KDCDUMP
/
/ " ------------------------------------------------------------------ "
/ "  If you want to do some actions after PEND ER, then you must      "
/ "   insert commands here. (E.g. prepare an UTM dump.)                "
/ " ------------------------------------------------------------------ "
/
*/        REMOVE-FILE-LINK LINK-NAME=KDCDUMP
*/        SKIP-COMMANDS TO-LABEL=EXEC
*/        SET-JOB-STEP
*/        REMARK TEXT=ACTIONS AFTER KDCAPPL PROG=NEW: *****
*/        REMARK TEXT==========================
*/.EXEC   REMARK
*/START-PROGRAM  *MODULE(  LIBRARY  = &START-LLM-LIB  -
*/                       , ELEMENT  = &START-LLM-NAME -
*/                       , RUN-MODE = *ADVANCED(                    -
*/                              UNRESOLVED-EXTRNS    = DELAY        -
*/                             ,LOAD-INFORMATION     = REFERENCES   -
*/                             )                                    -
*/                       , PROGRAM-MODE = *ANY                      -
*/                       )
/
*.UTM START FILEBASE  = &FILEBASE
/IF ( ROOTNAME <> '*NONE' )   " If dynamic root is wanted  ---------- "
*.UTM START ROOTNAME  = &ROOTNAME
*.UTM START TABLIB    = &TABLIB
/END-IF
/
*.UTM START STARTNAME = &STARTNAME
*.UTM START TASKS     = &TASKS
*.UTM START ASYNTASKS = &ASYNTASKS
*.UTM START TESTMODE  = &TESTMODE
*.UTM START PASSWORD  = &PASSWORD
*.UTM END
/
/ " ------------------------------------------------------------------ "
/ "  Here follows the section with the data base and format system     "
/ "   parameters (if necessary).                                       "
/ " ------------------------------------------------------------------ "
/
/
*.FHS MAPLIB          = &MAPLIB
*.UTM END
*/        SKIP-COMMANDS TO-LABEL=REPEAT
*/        SET-JOB-STEP
*/        SHOW-FILE-LINK LINK-NAME=KDCTRMAP
/
/ " ------------------------------------------------------------------ "
/ "  If you want to do some actions after abnormal application         "
/ "   termination, then you can put some commands here.                 "
/ " ------------------------------------------------------------------ "
/
*/        SKIP-COMMANDS TO-LABEL=LOGOFF
*/        SET-JOB-STEP
*/        REMARK TEXT=ACTIONS AFTER NORMAL APPLICATION TERMINATION:
*/        REMARK TEXT==========================
/
```

```
/ " ------------------------------------------------------------------ "
/ "  Here you can insert commands to execute after normal              "
/ "   application termination.                                         "
/ " ------------------------------------------------------------------ "
/
*/.LOGOFF EXIT-JOB SYSTEM-OUTPUT = *NONE
@CH ON
@DELETE &&:1-1:
@W'&STARTNAME'
@H
/MODIFY-JOB-SWITCHES OFF=(4,5)
/
/END-BLOCK EDT
/
/IF START
/   ENTER-JOB FROM-FILE=&STARTNAME
/   END-IF
/
/MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=USER-ACKNOWLEDGE
/EXIT-PROCEDURE
```

5.5 Fehlermeldungen beim Start

Wird der Start einer UTM-Anwendung oder einer Task wegen eines Fehlers abgebrochen, gibt *open*UTM in der Regel die Meldungen K049 und/oder K078 aus. Die Meldung K078 kann in mehreren Varianten auftreten. Die Bedeutung dieser Meldungen und deren Returncodes sind ausführlich im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“ erklärt.

Es wird zum nächsten Kommando /SET-JOB-STEP oder zum /LOGOFF-Kommando in der Prozedur verzweigt.

Startfehler können beim Start einer jeden Task auftreten.

5.6 Nach abnormalem Anwendungsende erneut starten

*open*UTM bietet die Möglichkeit, eine Anwendung nach einem abnormalen Anwendungsende automatisch neu starten zu lassen.

i Als Voraussetzung für das im Folgenden geschilderte Vorgehen wird das Produkt JV („Jobvariablen") benötigt.

Wird eine Jobvariable mit dem Basisnamen der A-Teile der KDCFILE (inklusive Catid und Userid) katalogisiert, so wird diese Jobvariable von *open*UTM benutzt. Die erste Spalte der Jobvariablen kann die Werte R oder T annehmen:

R: UTM-Anwendung läuft

T: UTM-Anwendung ist beendet

Die zweite Spalte der Jobvariablen kann die Werte N oder A annehmen:

N: normal beendet

A: abnormal beendet

Diese Jobvariable kann dazu benutzt werden, den Ablauf einer UTM-Anwendung durch einen ENTER-Job zu überwachen, der im Fall, dass sich die Anwendung abnormal beendet, die Anwendung neu startet. Wenn die erste Spalte der Jobvariablen auf "T" steht, wird abhängig vom Wert der zweiten Jobvariablenspalte entweder dieser ENTER-Job beendet (bei ‘N‘) oder die UTM-Anwendung neu gestartet (bei ‘A‘).

Der ENTER-Job kann folgenden Aufbau haben:

```
/.WAIT1 REMARK
/WAIT   ((jobvariable-name,1,1)='T'),TIME=32767,TIMELAB=.WAIT1
/SKIPJV ((jobvariable-name,2,1)='N'),LABEL=.END
/ENTER enterfile,TIME=ttt,JOB-CLASS=job-class
/WAIT ((jobvariable-name,1,1)='R')
/WAIT ,TIME=600,TIMELAB=.WAIT2
/.WAIT2 REMARK
/SKIPJV ((jobvariable-name,1,2)='TA'),LABEL=.END
/SKIP .WAIT1
/.END REMARK
/LOGOFF
```

Beschreibung des ENTER-Jobs

- Die erste WAIT-Anweisung lässt den Job warten, bis sich die UTM-Anwendung beendet hat (‘T‘ im ersten Feld der JV). Bei einer abnormalen Beendigung (‘A‘ im zweiten Feld der JV) wird der ENTER-Job zum erneuten Starten der UTM-Anwendung angestoßen, bei normaler Beendigung wird auch der Überwachungsjob beendet.
- Mit der zweiten WAIT-Anweisung wird gewartet, bis die UTM-Anweisung wieder läuft.
- Mit der dritten WAIT-Anweisung wird dann zehn Minuten (600 sek) gewartet, bis abgefragt wird, ob sich die UTM-Anwendung erneut abnormal beendet hat. Wenn ja, wird der Überwachungsjob beendet, und die UTM-Anwendung wird nicht hochgefahren. Wenn nicht, wird zum ersten WAIT zurückverzweigt, und der Überwachungsjob legt sich erneut auf die Lauer. Damit wird verhindert, dass die UTM-Anwendung bei einem permanenten Fehler ständig beendet und wieder hochgefahren wird.

6 UTM-Anwendung beenden

Eine UTM-Anwendung kann

– normal beendet werden durch Administrationsaktionen
– abnormal beendet werden durch Fehler oder per Administration

*open*UTM beendet alle Prozesse mit TERMJ, d.h. es wird zum nächsten /SET-JOB-STEP-Kommando oder zum /LOGOFF-Kommando in der Startprozedur verzweigt.

In der Startprozedur können Sie abfragen, ob die UTM-Anwendung normal oder nicht normal (TERMAP) beendet wurde. Siehe auch Abschnitt „Startparameter für openUTM“ auf Seite 82. Für die Abfrage gibt es folgende Möglichkeiten:

– Auftragsschalter 3
– TFT-Eintrag
– Jobvariable

6.1 UTM-Anwendung normal beenden

Eine UTM-Anwendung beendet der UTM-Administrator normal, indem er z.B. folgendes UTM-Administrationskommando an einem Administrationsterminal eingibt:

```
KDCSHUT NORMAL
```

oder

```
KDCSHUT WARN,TIME=time
```

In Anwendungen, die verteilte Transaktionsverarbeitung verwenden, sollte eine Anwendung immer mit KDCSHUT WARN beendet werden, denn dies erlaubt eine geordnete Beendigung der offenen verteilten Transaktionen.

Nach Eingabe des Kommandos führt *open*UTM die folgenden Aktionen durch:

– Es werden alle Aufträge abgearbeitet, die noch in der UTM-Warteschlange sind.
– Die Verbindungen zu Terminals werden abgebaut.
– KDCFILE, System-Protokolldatei und Benutzer-Protokolldatei werden in einen konsistenten Zustand gebracht und ordnungsgemäß geschlossen.
– Alle Prozesse der Anwendung werden beendet.

Um eine UTM-Anwendung normal zu beenden, können Sie an Stelle des Kommandos KDCSHUT auch die entsprechende Funktion bei WinAdmin oder an der Administrations-Programmschnittstelle verwenden.

Der Systemadministrator kann die UTM-Anwendung auch mit dem BCAM-Kommando BCAPPL normal beenden:

```
BCAPPL APPLICATION=anwendungsname,MODE=DEACTIVATE,W=NO
```

Nach Eingabe dieses Kommandos an der Konsole wird die Anwendung von BCAM geschlossen. *open*UTM kann offene Verbindungen zu den Kommunikationspartnern nicht mehr ordnungsgemäß schließen.

```
BCAPPL APPLICATION=anwendungsname,MODE=DEACTIVATE,W=YES
```

Nach Eingabe dieses Kommandos an der Konsole erhält die Anwendung von BCAM eine Warnung, dass sie von BCAM geschlossen wird. *open*UTM baut dann sofort offene Verbindungen ab und beendet sich. Die Anwendung ist daher u.U. beendet, bevor die in BCAM generierte Warnzeit abgelaufen ist.

Wird BCAM mit BCEND beendet, so wird eine zu diesem Zeitpunkt geladene UTM-Anwendung normal beendet.

Falls die UTM-Anwendung von einer Jobvariablen überwacht wurde (siehe Seite 95), wird die erste Spalte der Jobvariablen auf "T" und die zweite Spalte auf "N" gesetzt.



Mit dem Kommando /CANCEL-JOB lässt sich eine UTM-Task nur dann beenden, wenn sie im „nicht-privilegierten“ Zustand (TU) läuft. Mit diesem Kommando kann beispielsweise ein Teilprogramm in einer Endlos-Schleife beendet werden.

6.2 UTM-Anwendung abnormal beenden

Eine UTM-Anwendung kann durch folgende Ereignisse abnormal beendet werden:

- UTM-interner Fehler
- Fehler in der Systemumgebung, z.B. Datenbanksystem nicht verfügbar
- UTM-Administrationskommando KDCSHUT KILL (oder durch die entsprechende Funktion bei WinAdmin oder KDCADMI)
- Generierungsfehler

Bei abnormaler Beendigung werden folgende Aktionen ausgeführt:

- Alle Transaktionen, die gerade von den einzelnen Prozessen bearbeitet werden, werden abgebrochen.
- Die Verbindungen zu BCAM und damit allen Kommunikationspartnern der Anwendung werden abgebaut.

- Die Verbindung zum Datenbanksystem wird abgebaut.
- Für jede Task der Anwendung wird ein UTM-spezifischer Speicherauszug (UTM-Dump) erzeugt. Siehe dazu auch *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.
- Alle Prozesse der Anwendung werden beendet und alle Dateien werden geschlossen. Es wird nicht versucht, die KDCFILE in einen konsistenten Zustand zu bringen. Dies geschieht erst bei einem erneuten Start der Anwendung.

Falls die UTM-Anwendung von einer Jobvariablen überwacht wurde (siehe Seite 95), wird die erste Spalte der Jobvariablen auf "T" und die zweite Spalte auf "A" gesetzt.

Nach einem abnormalen Beenden der Anwendung sollten Sie als erstes die Ursache für den Abbruch feststellen. Dazu suchen Sie im SYSLST-Protokoll die Meldung K060. Diese Meldung enthält als Insert den Dump-Fehlercode, der genaue Auskunft über die Abbruch-Ursache gibt. Die Dump-Ursache können Sie auch als Teil des Dateinamens der UTM-Dumps finden. Die Bedeutung des Dump-Fehlercodes ist im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“ bei der Meldung K060 beschrieben. Dabei gibt es drei Möglichkeiten:

- Der Dump-Fehlercode sagt aus, dass ein KDCDEF-Operand verändert werden muss. Dann ist die KDCFILE neu zu generieren. Wenn Sie die Benutzerdaten im Pagepool erhalten wollen, gehen Sie wie folgt vor:
  - Warmstart mit ASYNTASKS=0, TASKS=1
  - Anwendung normal beenden mit KDCSHUT NORMAL
  - neue KDCDEF-Generierung mit dem veränderten Operanden
  - Übertragung der Benutzerdaten mit KDCUPD aus der alten in die neue KDCFILE
  - Start der Anwendung mit der neuen aktualisierten KDCFILE
- Der Dump-Fehlercode nennt als Ursache
  - DMS-Fehler
  - Hardware-Fehler eines Terminals
  - Datenbanksystem ist z.Zt. nicht verfügbar

  Ist der Fehler behoben, können Sie die Anwendung erneut starten, *open*UTM führt dann automatisch einen Warmstart durch.
- Es liegt ein Systemfehler vor. In diesem Fall erstellen Sie Diagnoseunterlagen und schreiben eine Fehlermeldung an den Systembetreuer. Die Unterlagen, die der Systembetreuer benötigt, sind im Abschnitt „Diagnoseunterlagen für eine Problemmeldung“ auf Seite 100 zusammengestellt.

  Ein Warmstart mit derselben KDCFILE ist in diesem Fall nicht immer erfolgreich. Kann kein Warmstart durchgeführt werden, müssen Sie die KDCFILE mit KDCDEF neu generieren.

6.3 Diagnoseunterlagen für eine Problemmeldung

Beim abnormalen Ende einer UTM-Anwendung durch einen Systemfehler schreibt *open*UTM Dumps für alle Tasks der Anwendung, siehe *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.

Folgende Diagnoseunterlagen sind unbedingt mitzuliefern, wenn Sie eine Problemmeldung für den Systemkundendienst schreiben:

– UTM-Dump-Dateien von allen Tasks
– SYSOUT-Protokolle von allen Tasks
– System-Protokolldatei SYSLOG
– KDCDEF-Steueranweisungen

UTM-Dumps und SYSLOG-Datei sollten nicht aufbereitet werden.

Zusätzlich sollten Sie für eventuelle Nachforderungen folgende Unterlagen sicherstellen:

– KDCFILE
– das Protokoll vom Binden des Anwendungsprogramms

7 UTM-Datenbank-Anwendung

Dieses Kapitel gibt einen zusammenhängenden Überblick über das, was für den Einsatz von Datenbanken unter *open*UTM zu beachten ist.

*open*UTM kann mit folgenden Datenbanksystemen koordiniert zusammenarbeiten:

- UDS/SQL
- SESAM/SQL
- ORACLE
- CIS
- PRISMA
- LEASY (das Dateisystem LEASY verhält sich gegenüber *open*UTM wie ein Datenbank-System)

Darüber hinaus kann *open*UTM auch mit anderen Datenbanksystemen koordiniert zusammenarbeiten, wenn diese die Schnittstelle zur UTM-DB-Zusammenarbeit (IUTMDB) unterstützen.

Multi-DB-Betrieb

*open*UTM kann auch koordiniert mit 2 Datenbanksystemen unterschiedlichen Typs zusammenarbeiten, d.h in einer UTM-Transaktion dürfen Aufrufe an beide Datenbanksysteme enthalten sein. Wenn dabei von beiden Systemen Update-Operationen angefordert werden, dann muss mindestens eines der beiden Datenbanksysteme das „2-Phase-Commit“-Protokoll unterstützen.

Für UDS/SQL und SESAM/SQL ist die koordinierte Zusammenarbeit von *open*UTM mit den beiden Systemen "in *einer* Multi-DB-Anwendung" gewährleistet, wenn

- für Aufrufe an UDS/SQL die COBOL- oder Call-DML
- und für Aufrufe an SESAM/SQL die SQL benutzt wird.

Die Systeme UDS/SQL und SESAM/SQL können jedoch nicht in einer Anwendung kombiniert werden, falls für die Aufrufe an das DB-System UDS/SQL die SQL verwendet wird.

Auf Sonderfreigabe ist Multi-DB-Betrieb mit mehr als 2 DB-Systemen möglich (derzeit bis zu 8 DB-Systeme).



Nähere Information über das Konzept der koordinierten Zusammenarbeit finden Sie im *open*UTM-Handbuch „Konzepte und Funktionen“. Weiterführende Details sind in den Handbüchern zu den DB-Systemen unter dem Thema UTM bzw. *open*UTM beschrieben, z.B. „UDS/SQL - Anwendungen programmieren“ oder „SESAM/SQL - Basishandbuch“.

7.1 UTM-Datenbank-Anschluss generieren

Für eine koordinierte Zusammenarbeit müssen Sie den UTM-Datenbank-Anschluss in der KDCDEF-Anweisung DATABASE generieren. Dort geben Sie an:

- den Typ des Datenbanksystems

  Für fremde Datenbanksysteme muss der Typ „DB“ angegeben werden.

- den Entry-Namen der Datenbank
- die Bibliothek, aus der das Datenbank-Verbindungsmodul nachgeladen werden kann.

  Für die Bibliothek können Sie einen symbolischen Link-Namen angeben. Dadurch wird bei den Datenbanksystemen SESAM/SQL und UDS/SQL das Verbindungsmodul im IMON-Installationspfad gesucht und aus der Bibliothek nachgeladen. Dies hat den Vorteil, dass die UTM-Anwendung unabhängig ist von Installationskennungen und Bibliotheksnamen des Datenbanksystems.

Die Anweisung DATABASE kann mehrfach angegeben werden. Wenn Sie zwei Datenbanksysteme vom gleichen Typ definieren, dann müssen sich die Entry-Namen unterscheiden. Wenn Sie zwei Datenbanksysteme unterschiedlichen Typs definieren, dann können die DATABASE-Anweisungen in beliebiger Reihenfolge gegeben werden. Dabei können alle Datenbanksysteme kombiniert werden, die man bei der Generierung angeben kann; eines der Systeme darf auch ein Fremdsystem sein (DATABASE=DB).



Näheres finden Sie im *open*UTM-Handbuch „Anwendungen generieren“ bei der Beschreibung der DATABASE-Anweisung.

ROOT-Source erzeugen und assemblieren

Beim KDCDEF-Lauf wird eine ROOT-Source erzeugt, die entsprechend den Angaben bei DATABASE folgende Makroaufrufe enthält:

- Bei Betrieb mit einem Datenbanksystem der Makroaufruf KDCDB.
- Bei Betrieb mit mehreren Datenbanksystemen die Makroaufrufe KDCDB*x*. Dabei ist *x* der erste Buchstabe des in DATABASE angegebenen Datenbank-Typs, z.B. KDCDBS für SESAM.

Die Makro KDCDB bzw. die Makros KDCDBx werden mit den jeweiligen Datenbanksystemen ausgeliefert. Beim Assemblieren der ROOT-Source müssen Sie die Bibliotheken, in der diese Makros zu finden sind, explizit zuweisen.



Näheres zur Erzeugen der Root-Source finden Sie im *open*UTM-Handbuch „Anwendungen generieren“ im Kapitel „Anwendungskomponenten generieren“. Das Erzeugen von Lademodulen ist im Abschnitt „Lademodule generieren“ auf Seite 15 beschrieben.

7.2 UTM-Datenbank-Anwendung binden

*open*UTM benötigt für die koordinierte Zusammenarbeit mit einer Datenbank ein Datenbank-Verbindungsmodul. Dieses Modul wird mit dem betreffenden Datenbanksystem ausgeliefert. Die Bibliothek, in der dieses Verbindungsmodul steht, ist eine optionale Angabe in der KDCDEF-Anweisung DATABASE (siehe oben).

Das Verbindungsmodul kann wahlweise dynamisch nachgeladen werden oder beim Binden der Anwendung mit eingebunden werden. Wenn *open*UTM mit zwei Datenbanksystemen zusammenarbeiten soll, wird der für jedes Datenbanksystem spezifische Verbindungsmodul benötigt.

Verbindungsmodul dynamisch nachladen

Der Verweis auf den Verbindungsmodul kann beim Binden des Anwendungsprogramms weggelassen werden. Dies hat den Vorteil, dass das Anwendungsprogramm beim Wechsel auf eine neue Datenbankversion (mit neuem Verbindungsmodul) nicht neu gebunden werden muss. In diesem Fall wird das Verbindungsmodul beim Start der Anwendung dynamisch nachgeladen. Dabei gilt der folgende Suchalgorithmus:

1. im Link-Kontext
2. im Shared Code des Benutzers
3. in nicht-privilegierten Subsystemen (z.B. kann das DB-System als Subsystem geladen werden)
4. im Shared Code des Systemadressraums
5. in der Bibliothek, die beim Generieren in der DATABASE-Anweisung angegeben wurde
6. in den alternativen Bibliotheken, die Sie beim Anwendungsstart mit einem SET-FILE-LINK-Kommando über die Linknamen BLSLIB*nn* ($00 \leq nn \leq 99$) bekanntgemacht haben.

Das heißt insbesondere, dass das Verbindungsmodul **nicht** aus der in der DATABASE-Anweisung angegebenen Bibliothek geladen wird, falls das DB-System vollständig als nicht-privilegiertes Subsystem geladen wurde oder falls Teile des DB-Systems als nicht-privilegierte Subsysteme geladen wurden und der Verbindungsmodul in einem dieser Teile enthalten ist.



ACHTUNG!
Beim Nachladen ist zu beachten, dass das Verbindungsmodul des Datenbanksystems gegebenenfalls nur im 24-Bit-Modus ablauffähig ist. In diesem Fall muss beim Binden der UTM-Anwendung die Anweisung MODIFY-SYMBOL-ATTRIBUTES ... ADDRESSING-MODE=24 (BINDER) eingefügt werden.

Verbindungsmodul einbinden

Wenn das Verbindungsmodul mit eingebunden wird, dann sollte man die INCLUDE-MODULES-Anweisung verwenden. Die RESOLVE-BY-AUTOLINK-Anweisung kann zu unerwünschten Nebeneffekten führen, wenn in den bei RESOLVE-BY-AUTOLINK angegebenen Bibliotheken außer dem Verbindungsmodul noch andere Module enthalten sind.

7.3 UTM-Datenbank-Anwendung starten und beenden

Eine UTM-Datenbank-Anwendung wird wie eine UTM-Anwendung ohne Datenbank gestartet und beendet, d.h. durch Starten und Beenden des UTM-Anwendungsprogramms.

Dabei sollten Sie beachten, dass der DBH der Datenbank immer **vor** der UTM-Anwendung gestartet wird, da es andernfalls einen Startfehler bei *open*UTM geben kann.

Startparameter für eine UTM-Datenbank-Anwendung

Für den Start einer UTM-DB-Anwendung müssen neben den Startparametern für *open*UTM auch die Startparameter für die Datenbank(en) angegeben werden, falls das DB-System dies fordert. Diese Startparameter sind datenbank-spezifisch und werden in den Handbüchern des jeweiligen DB-Systems beschrieben.

Die Startparameter müssen nach folgendem Schema eingegeben werden, wobei die Startparameter von *open*UTM immer **vor** den Startparametern der Datenbank stehen:

`.UTM`	Startparameter für UTM, siehe Abschnitt „Startparameter der Anwendung“ auf Seite 81; eventuell muss der Parameter DBKEY angegeben werden.
	...
.db-typ1	Startparameter für die Datenbank vom Typ *db-typ1*, siehe Handbuch für das DB-System *db-typ1*.
.db-typ2	Startparameter für die Datenbank vom Typ *db-typ2*, wenn ein zweites DB-System eingesetzt wird, siehe Handbuch für das DB-System *db-typ2*.
	...

Für *db-typ1*, *db-typ2* müssen Sie den Datenbank-Typ angeben, den Sie in der DATABASE-Anweisung generiert haben.

Für UDS können Sie z.B. angeben:

```
.UDS DATABASE=UDSCONF
```

Damit wird der Anschluss der UTM-Anwendung an die UDS-Konfiguration UDSKONF festgelegt.



Bei SESAM/SQL ab V2.2 müssen die Startparameter in einer Konfigurationsdatei bereitgestellt werden, die den Tasks der UTM-SESAM-Anwendung unter dem Linknamen SESCONF zugewiesen wird.

UTM-Datenbank-Anwendung normal beenden

Eine UTM-Datenbank-Anwendung beenden Sie mit Hilfe der UTM-Administration, siehe Abschnitt „UTM-Anwendung normal beenden“ auf Seite 97. Der Database Handler wird dabei nicht heruntergefahren.

UTM-Datenbank-Anwendung abnormal beenden

Eine UTM-Datenbank-Anwendung kann per Administration oder aufgrund von Fehlern abnormal beendet werden, siehe Abschnitt „UTM-Anwendung abnormal beenden“ auf Seite 98. Nach einem abnormalen Anwendungsende kann es vorkommen, dass die Datenbank und KDCFILE in einem nicht-konsistentem Zustand sind.

In diesem Fall wird die Konsistenz der Daten beim anschließenden Warmstart der UTM-Datenbank-Anwendung überprüft und ggf. wieder hergestellt. Dabei führt *open*UTM eine gemeinsame Recovery Phase mit den beteiligten Datenbanksystemen durch.

7.4 Betrieb einer UTM-Datenbank-Anwendung

Der Betrieb einer UTM-Datenbank-Anwendung gehorcht den selben Prinzipien wie der Betrieb einer UTM-Anwendung ohne Datenbank. Die Besonderheiten, die dabei zu beachten sind, sind in den nachfolgenden Abschnitten beschrieben.

7.4.1 Anmelden und Abmelden eine Benutzers

Ein Benutzer, der mit einer UTM-Datenbank-Anwendung arbeiten möchte, meldet sich über die client-spezifischen Anmeldeverfahren bei *open*UTM an. Entsprechendes gilt für das Abmelden.

Wenn zum Anmelden ein Anmeldevorgang benützt wird, dann ist Folgendes zu beachten:

- Meldet sich der Benutzer über ein Terminal oder eine Terminalemulation an, dann sind Datenbank-Aufrufe im ersten Teil des SIGNON-Services aus Sicherheitsgründen nicht erlaubt, es sei denn, man lässt dies per Generierung explizit zu mit der KDCDEF-Anweisung SIGNON, ...RESTRICTED=NO.
- Im zweiten Teil des Anmeldevorgangs kann das Berechtigungsprofil für den Benutzer aus der Datenbank gelesen werden, sofern die Datenbank diese Möglichkeit unterstützt. Damit lässt sich ein DB/DC-übergreifendes Berechtigungskonzept realisieren.

Näheres zum An- und Abmelden finden Sie im Kapitel „Arbeiten mit einer UTM-Anwendung“ auf Seite 111.

7.4.2 SAT-Protokollierung

Die SAT-Protokollierung kann für jeden Transaktionscode (= auftragsgesteuert) und jeden UTM-Benutzer (= benutzergesteuert) einzeln definiert werden. UTM protokolliert dann alle sicherheitsrelevanten Ereignisse innerhalb von UTM und setzt an der Schnittstelle zum DB-System eine Anzeige, dass die SAT-Protokollierung aktiviert wurde.

Sicherheitsrelevante Ereignisse innerhalb der DB-Transaktionen müssen vom DB-System, protokolliert werden, falls dies vorgesehen ist. UTM protokolliert nur, ob die Transaktion als Ganzes erfolgreich ausgeführt wurde oder nicht.

Näheres zur SAT-Protokollierung finden Sie in Abschnitt „Benutzergesteuerte SAT-Protokollierung“ auf Seite 161 und in Abschnitt „Auftragsgesteuerte SAT-Protokollierung“ auf Seite 161.

7.4.3 Accounting

DB-Systeme, die nicht selbst in UTM-Tasks ablaufen, können *open*UTM den Betriebsmittelverbrauch für die Bearbeitung von DB-Aufrufen über die IUTMDB-Schnittstelle mitteilen. Die vom DB-System gelieferten Verbrauchswerte (CPU, I/Os) werden in der Kalkulationsphase im Kalkulationssatz abgelegt, siehe Abschnitt „Aufbau des Kalkulationssatzes“ auf Seite 254.

Diese Werte werden auch an SM2 geliefert. Da diese Daten u.U. vom DB-System nicht exakt ermittelt werden, ist ihre Auswertung zu Messzwecken nur eingeschränkt möglich, Tendenzen sind jedoch feststellbar. Zum Erfassen exakter Werte vom DB-System steht der Messmonitor KDCMON zur Verfügung, siehe unten.

In der Abrechnungsphase werden sie mit den festgelegten Gewichten multipliziert und auf den Verrechnungseinheitenzähler addiert, siehe Abschnitt „Aufbau des Abrechnungssatzes“ auf Seite 253.

Die Verbrauchswerte werden von den DB-Systemen SESAM/SQL und UDS/SQL an *open*UTM geliefert. SESAM und SESAM/SQL liefern die Verbrauchswerte nur dann an *open*UTM, wenn auch im DBH die Sammlung von Abrechnungsinformationen eingeschaltet ist, siehe dazu Handbuch „Datenbankbetrieb SESAM/SQL“.

7.4.4 Leistungskontrolle

Zur Leistungskontrolle der UTM-Anwendung kann der UTM-Messmonitor KDCMON mit dem Aufbereitungs-Tool KDCEVAL eingesetzt werden. Einige der mit KDCEVAL aufbereiteten Listen enthalten auch datenbankspezifische Informationen:

- Die Liste TASKS enthält in der Spalte `database` den Zeitanteil für DB-Aufrufe.
- Die Liste KCOP gibt in NOOP an, wie oft DB-Messdaten geschrieben wurden.
- Die Liste TACLIST gibt TAC-spezifisch die Anzahl der DB-Aufrufe an.
- Die Listen TRACE und TRACE2 sind Trace-Listen, die sowohl die UTM-Aufrufe als auch die DB-Aufrufe enthalten.

Näheres zu KDCMON und den zugehörigen Auswertungslisten finden Sie im Abschnitt „UTM-Messmonitor KDCMON“ auf Seite 196f.

7.4.5 Diagnose

Wichtige Informationsquellen zur Fehlerdiagnose einer UTM-Anwendung sind UTM-Meldungen, Fehlercodes und Dumps. Bei einer UTM-Datenbank-Anwendung findet man darin zusätzlich datenbankspezifische Informationen. Diese sollten als erstes herangezogen werden, wenn ein Fehler mit der Datenbank-Kopplung zusammenhängen könnte. Dabei handelt es sich um folgende UTM-Diagnose-Informationen:

- Die datenbankspezifischen UTM-Meldungen K068 und K071
- Die Start-Fehlercodes der Meldung K049
- Den inkompatiblen Returncode KCRCDC
- Die DB-DIAGAREA des UTM-Dumps, falls ein UTM-Dump erzeugt wurde
- Die CDUMP-Fehlercodes KDCDBxx, falls ein CDUMP erzeugt wurde.

Einzelheiten dazu finden Sie im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.

8 Arbeiten mit einer UTM-Anwendung

Dieses Kapitel beschreibt, wie ein Benutzer über die unterschiedlichen Clients mit einer UTM-Anwendung kommunizieren kann. Für alle Clients verläuft die Kommunikation immer nach folgendem Prinzip:

1. Anmelden an die UTM-Anwendung
   Ein Benutzer kann sich nur über Clients anmelden, für die LTERM-Partner, LTERM-Pools oder als OSI-LPAP-Partner in der UTM-Anwendung generiert sind, siehe *open*UTM-Handbuch „Anwendungen generieren“. Eine Anmeldung z.B. über Remote-Login-Mechanismen ist nicht möglich.
2. Services (=Vorgänge) der UTM-Anwendung aufrufen:
   Für den Zugriffsschutz bietet *open*UTM ein eigenes Berechtigungskonzept, siehe Seite 136.
3. Ggf. UTM-Benutzerkommandos eingeben.
4. Abmelden von der UTM-Anwendung

Im Detail unterscheiden sich diese Schritte je nach Art des Clients. Die folgenden Abschnitte beschreiben, welche Möglichkeiten es für die verschiedenen Clients gibt.

Für den Zugang werden UTM-Benutzerkennungen verwendet, sofern die Anwendung mit Benutzerkennungen generiert ist. Das Anmelden an eine UTM-Anwendung ohne Benutzerkennungen wird auf Seite 132 behandelt.

8.1 Anmeldeverfahren mit Benutzerkennungen

Ist eine Anwendung mit Benutzerkennungen generiert, dann führt *open*UTM für den Benutzer abhängig von der Art des Client ein Standard-Anmeldeverfahren durch. Daneben besteht die Möglichkeit, anstelle des Standard-Verfahrens selbst erstellte Anmeldeverfahren zu verwenden, siehe Abschnitt „Anmeldeverfahren mit Anmeldevorgängen“ auf Seite 125.

Ein Benutzer kann sich über folgende Client-Zugänge anmelden:

– Terminals (Seite 112)

– UPIC-Clients und TS-Anwendungen (Seite 120)

– OSI-TP-Clients (Seite 122)

– über das Web mit Hilfe von Web*Transactions* (Seite 123)

Grundsätzlich können sich auch mehrere Benutzer unter einer Benutzerkennung anmelden, siehe Abschnitt „Mehrfach-Anmeldungen unter einer Benutzerkennung“ auf Seite 124.

8.1.1 Standard-Anmeldeverfahren für Terminals

Der Benutzer führt folgende Schritte durch, um über ein Terminal mit einer UTM-Anwendung zu arbeiten:

1. Verbindungsaufbau zur UTM-Anwendung

   Die Verbindung kann entweder automatisch durch *open*UTM beim Anwendungsstart aufgebaut werden oder manuell durch den Benutzer.

   Die Art des Verbindungsaufbaus wird bei der KDCDEF-Generierung festgelegt. Vor dem Aufbau der Verbindung muss selbstverständlich eine physische Verbindung zum TRANSDATA-Netz bestehen.

2. Anmelden bei einer UTM-Anwendung, siehe Abschnitt „Standard-Anmeldedialog“ auf Seite 114.

Erst dann kann der Benutzer Vorgänge starten und im Dialog bearbeiten, siehe Abschnitt „UTM-Services aufrufen“ auf Seite 133.

Automatischer Verbindungsaufbau

Der automatische Verbindungsaufbau ist nur für explizit generierte Terminals möglich. Wenn das Terminal beim Start der UTM-Anwendung eingeschaltet und zu diesem Zeitpunkt mit keiner anderen Anwendung verbunden ist, dann bestätigt *open*UTM den Verbindungsaufbau mit der Meldung K002.

Expliziter Verbindungsaufbau über Terminal-Emulation

Ein Benutzer kann auch von einem Windows PC aus mit einer UTM-Anwendung arbeiten, indem er eine Emulation wie z.B. MT9750 aufruft und die Sitzung startet, die für diese Anwendung konfiguriert ist. Der Benutzer gibt beim Start der Sitzung nur den Sitzungstitel an.

Wenn Sie eine Sitzung für MT9750 einrichten, dann geben Sie den Namen der UTM-Anwendung sowie den Namen und/oder die IP-Adresse des Rechners an, auf dem die Anwendung läuft. Details dazu finden Sie im MT9750 Produkthandbuch.

Expliziter Verbindungsaufbau in TRANSDATA-Netzen

Der Benutzer stellt die Verbindung zur UTM-Anwendung selbst her. Auf die Aufforderung:

```
PLEASE ENTER NET COMMAND
```

gibt der Benutzer das folgende Kommando ein:

O *applicationname*[,*processornumber*/*regionsnumber*][,PW=*connection-password*]

Bedeutung der Parameter:

applicationname

Name der Anwendung mit der der Benutzer verbunden werden will. Die Angabe von *applicationname* kann entfallen, wenn bei der PDN-Generierung ein Verbindungsvorschlag generiert wurde.

processornumber/regionsnumber

processornumber/*regionsnumber* beziehen sich auf das Datenkommunikationssystem TRANSDATA. *processornumber*/*regionsnumber* des Rechners, zu dem das Terminal Verbindungen unterhalten soll, kann bei der Generierung des Datenkommunikationssystems bereits festgelegt (vordefiniert) werden. In diesem Fall entfällt die Angabe von *processornumber*/*regionsnumber*.

Format von *processornumber*/*regionsnumber*:

```
ppp/rrr
```

ppp: Prozessornummer ($0 \leq ppp < 256$)
rrr: Regionsnummer ($0 \leq rrr < 256$)

connection-password

Ob ein Passwort zum Aufbau der Verbindung mit der UTM-Anwendung anzugeben ist, hängt von den Startparametern der UTM-Anwendung ab (siehe auch Seite 81). Die Informationen erhält der Benutzer vom UTM-Administrator.

Format von *connection-password*:

`C'`*aaaa*`'` oder `X'`*xxxxxxxx*`'`

aaaa: 4 abdruckbare Zeichen
xxxxxxxx: 8-stellige Sedezimalzahl

Wenn der Verbindungsaufbau durch den Benutzer nicht akzeptiert wird, erfolgt eine Rückmeldung ans Terminal.

Die Meldungen und ihre Bedeutung können dem TRANSDATA-Benutzerhandbuch „Netzzugang für Datenstationen“ entnommen werden.

8.1.1.1 Standard-Anmeldedialog

Der Standard-Anmeldedialog wird immer dann durchgeführt, wenn die beiden folgenden Bedingungen erfüllt sind:

1. Für das Terminal ist kein automatisches KDCSIGN (= automatische Berechtigungsprüfung) generiert (siehe Seite 120).
2. Für den Anwendungsnamen, unter dem sich der Benutzer angemeldet hat, ist kein Anmeldevorgang generiert (siehe Seite 125).

Beim Standard-Anmeldedialog führt *open*UTM eine Berechtigungsprüfung durch (Zugangskontrolle). Dabei wird der Benutzer über UTM-Meldungen im Zeilenmodus aufgefordert, sich zu legitimieren. Man kann bei der Generierung zwischen verschiedenen Varianten wählen, auch die Meldungstexte können dabei verändert werden. Darüberhinaus ist dieses Verfahren jedoch nicht modifizierbar.

Die Berechtigungsprüfung kann unterschiedlich streng festgelegt werden. Eine Übersicht aller Möglichkeiten enthält die Abbildung in Abschnitt „Szenarien für die UTM-Berechtigungsprüfung“ auf Seite 117.

*open*UTM fordert den Benutzer mit folgender Meldung zur Berechtigungsprüfung auf:

```
K002 Verbunden mit der Anwendung beispiel - Bitte KDCSIGN
```

Der Benutzer muss seine Berechtigung mit der folgenden Eingabe nachweisen:

KDCSIGN *kennung* [,*password*]

Hierbei ist *password* anzugeben, wenn ein Passwort ohne Dunkelsteuerung generiert ist (KDCDEF-Anweisung USER...,PASS=).

*open*UTM prüft die Benutzerkennung und ggf. das Passwort. Ist das Ergebnis negativ, dann wird der Benutzer zur Wiederholung der KDCSIGN-Eingabe aufgefordert.

Dunkelsteuerung des Passworts

Wenn für die Benutzerkennung Dunkelsteuerung des Passworts generiert ist (USER...,PASS=(*password*,DARK)), dann fordert *open*UTM den Benutzer mit einer Meldung auf, das Passwort in ein dunkel gesteuertes Feld einzugeben.

Bei jeder UTM-Anwendung hat der Benutzer die Möglichkeit, ein neues Passwort einzugeben, welches das bisherige ersetzen soll, vorausgesetzt, die minimale Gültigkeitsdauer erlaubt die Passwortänderung zu diesem Zeitpunkt. Das neue Passwort ist hierbei durch eine nochmalige Eingabe zu bestätigen. *open*UTM prüft das angegebene alte Passwort und ggf. das neue Passwort. Ist das alte Passwort nicht richtig oder stimmen die Angaben zum neuen Passwort nicht überein, dann wird der Benutzer mit einer UTM-Meldung informiert und zur Wiederholung der Eingabe aufgefordert.

Gültigkeitsdauer des Passwortes

Bei der Generierung der Benutzerkennung kann eine maximale und eine minimale Gültigkeitsdauer für das Passwort vereinbart werden (USER ...,PROTECT-PW=(...,*maxtime*,*mintime*)). Die minimale Gültigkeitsdauer bedeutet, dass der Benutzer nach einer Passwort-Änderung die nächste Änderung erst nach Ablauf der festgelegten Zeit vornehmen kann. Die maximale Gültigkeitsdauer bedeutet, dass der Benutzer das Passwort jeweils innerhalb der angegebenen Zeitspanne ändern muss.

Wird das Passwort innerhalb der nächsten 14 Tage nach der Anmeldung ungültig, warnt *open*UTM den Benutzer mit einer K-Meldung, sofern die minimale Gültigkeitsdauer des Passworts eine Änderung zu diesem Zeitpunkt erlaubt. Ein dunkelgesteuertes Passwort kann der Benutzer wie unter „Dunkelsteuerung des Passworts“ beschrieben ändern.
Um zu verhindern, dass Benutzer, die längere Zeit nicht mit der Anwendung arbeiten, die Passwort-Änderung versäumen und sich dann an den Administrator wenden müssen, kann die UTM-Anwendung so konfiguriert werden, dass auch nach Ablauf des Passworts eine Anmeldung erlaubt wird, siehe Abschnitt „Grace-Sign-On“ auf Seite 116.

*open*UTM überprüft bei der Passwortänderung, ob:

- das neue Passwort sich von dem alten Passwort unterscheidet.
  Wenn eine Passwort-Historie generiert ist (SIGNON ...,PW-HISTORY=*n*) und das Passwort mit beschränkter Gültigkeitsdauer (USER ...,PROTECT-PW=) generiert ist, dann wird auch gegen die letzten *n* Passworte geprüft.
- das neue Passwort der Komplexitätsstufe entspricht, die für die Benutzerkennung generiert wurde (USER ...,PROTECT-PW=),
- die Länge des Passwortes größer oder gleich der generierten Mindestlänge ist (USER ...,PROTECT-PW=)..

Erfüllt das Passwort alle diese Punkte, nimmt *open*UTM die Passwortänderung vor. Die Gültigkeitsdauer des neuen Passwortes entspricht wieder dem generierten Wert.
Erfüllt das neue Passwort einen dieser Punkte nicht, so wird der Benutzer mit der folgenden Meldung aufgefordert, die KDCSIGN-Eingabe mit dem alten Passwort zu wiederholen:

```
K097 Angaben zum neuen Passwort sind nicht verwendbar - Bitte KDCSIGN
```

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist kein Grace-Sign-On generiert, so wird die Anmeldung mit der folgenden Meldung zurückgewiesen:

```
K120 Die Gueltigkeit des Passworts ist abgelaufen
```

Diese Benutzerkennung kann sich erst wieder bei der UTM-Anwendung anmelden, wenn der UTM-Administrator der Kennung mit dem Kommando KDCUSER ein neues Passwort zugewiesen hat.

Grace-Sign-On

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist die Anwendung ist mit Grace-Sign-On generiert (SIGNON ...,GRACE=YES), so wird der Benutzer mit einer K-Meldung darauf hingewiesen, dass sein Passwort nicht mehr gültig ist. Gleichzeitig wird er aufgefordert, sein bisheriges Passwort und eine neues Passwort einzugeben. Zum Ändern muss auch ein hell generiertes, bereits eingegebenes Passwort nochmals in das entsprechende dunkelgesteuerte Feld eingegeben werden.

KDCSIGN mit Ausweiskarte

Wenn für die Benutzerkennung per Generierung das Einlegen einer Ausweiskarte (Magnetstreifenkarte) vorgeschrieben ist, fordert *open*UTM den Benutzer mit einer Meldung auf, eine Ausweiskarte in den Ausweisleser einzulegen. Zeitgleich zu dieser Meldung wird die Tastatur des Terminals gesperrt. Die Sperre wird erst durch Einlegen des Ausweises aufgehoben. Ist kein Ausweis verfügbar, so kann die Tastatur ersatzweise mit der Taste K14 oder ESC : entsperrt werden.

*open*UTM prüft die Ausweisinformation, d.h. *open*UTM vergleicht die in der KDCDEF-Anweisung USER ...,CARD=(...) definierte Zeichenfolge mit der entsprechenden Zeichenfolge auf dem Ausweis. Fällt das Ergebnis negativ aus, so wird der Benutzer mit der Meldung K031 informiert und zur Wiederholung der KDCSIGN-Eingabe aufgefordert. Fällt das Ergebnis positiv aus, dann:

- kann der Benutzer mit der Anwendung arbeiten, sofern kein dunkelgesteuertes Passwort für die Benutzerkennung generiert ist.
- fordert *open*UTM den Benutzer mit einer Meldung auf, das Passwort in ein dunkelgesteuertes Feld einzugeben. Die Anmeldung wird dann fortgesetzt, wie unter „Dunkelsteuerung des Passwortes“ bzw. „Gültigkeitsdauer des Passwortes“ beschrieben.

Der Ausweis muss in beiden Fällen solange im Ausweisleser bleiben, wie der Benutzer unter dieser Benutzerkennung arbeiten möchte. Wird er früher herausgezogen, so erkennt *open*UTM dies spätestens bei der nächsten Dialogeingabe und erzeugt implizit KDCOFF. Sehen Sie dazu auch Abschnitt „Abmelden von der UTM-Anwendung“ auf Seite 138.



Das Herausziehen des Ausweises erzeugt eine Nachricht entsprechend dem Drücken der Funktionstaste K14. Arbeitet der Benutzer unter einer Benutzerkennung, die KDCSIGN mit Ausweisprüfung verlangt, so hat die Zuordnung von K14 (KDCDEF-Anweisung SFUNC) zu einem TAC oder Returncode keinen Sinn. Die einzige Ausnahme bildet die Zuweisung von K14 zu KDCOFF, da die Eingabe von K14 zur Tastaturentsperrung sowieso KDCOFF auslöst.

Szenarien für die UTM-Berechtigungsprüfung

Das folgende Diagramm zeigt, welche Varianten der UTM-Berechtigungsprüfung möglich sind - abhängig von der KDCDEF-Generierung. Bei fehlerhaften Eingaben gibt *open*UTM eine spezifische Meldung aus und fordert den Benutzer zu einer neuen Eingabe auf. Werden von einem bestimmten Terminal aus nacheinander mehrere erfolglose Anmeldeversuche unternommen, dann erzeugt *open*UTM die Meldung K094 mit dem Standardziel SYSLOG (System-Protokolldatei). Die maximal zulässige Anzahl erfolgloser Anmeldeversuche bis zum Auslösen der Meldung K094 kann mit SIGNON ... SILENT-ALARM= bei der KDCDEF-Generierung festgelegt werden. Ein MSGTAC-Teilprogramm kann auf diese Meldung reagieren.

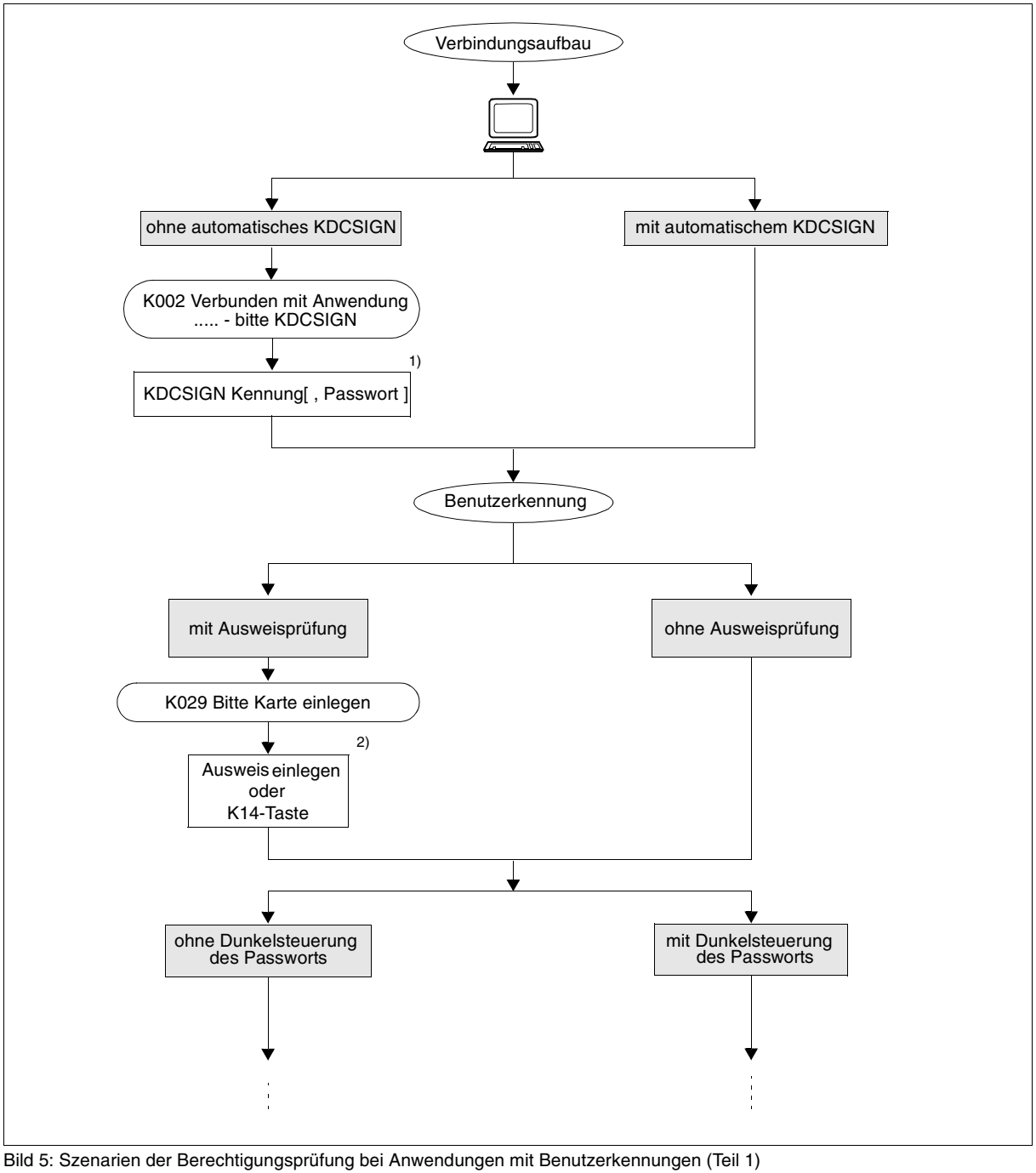
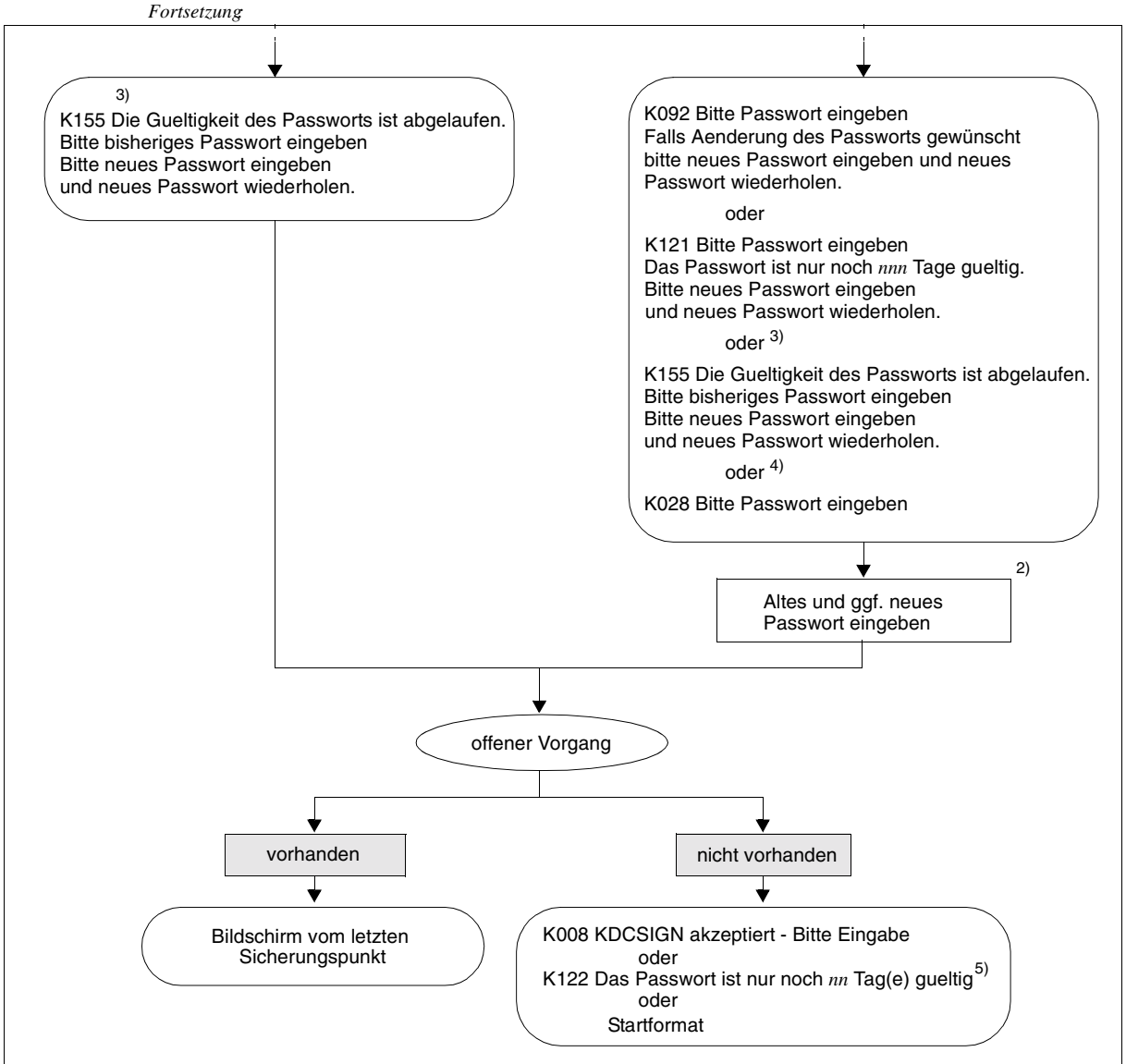


Bild 5: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 1)



[1] Akzeptiert *open*UTM die Eingabe nicht, weil falsche Kennung oder falsches Passwort eingegeben wurde bzw. Passwort abgelaufen ist, erfolgt Aufforderung zu neuem KDCSIGN.

[2] Akzeptiert *open*UTM die Eingabe nicht, erfolgt Aufforderung zu neuem KDCSIGN.

[3] Falls das Passwort abgelaufen und eine Änderung erlaubt ist; bei hellem Passwort nur bei Grace-Sign-On.

[4] Falls Änderung des Passworts nicht erlaubt ist.

[5] Bei Benutzern mit Passwort, wenn das Passwort weniger als 14 Tage gültig ist und wenn Änderung erlaubt ist und wenn kein Startformat für den Benutzer generiert ist

Bild 5: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 2)

8.1.1.2 Automatisches KDCSIGN

Wenn für ein Terminal die KDCDEF-Anweisung LTERM...,USER=*username* angegeben wurde, dann verhält sich *open*UTM nach dem Verbindungsaufbau so, als ob der Benutzer seine Berechtigung schon erfolgreich nachgewiesen hätte, d.h. er muss **kein** KDCSIGN eingeben. Wenn für diese Benutzerkennung die Eingabe einer Ausweiskarte oder eines dunkelgesteuerten Passworts vorgeschrieben ist, dann fordert *open*UTM diese Eingabe(n) vom Benutzer an.

Nach Eingabe von KDCOFF BUT kann man an diesem Terminal auch unter einer anderen Kennung arbeiten (siehe Abschnitt „Abmelden von der UTM-Anwendung“ auf Seite 138).

8.1.2 Anmeldeverfahren für UPIC-Clients und TS-Anwendungen

UPIC-Clients und TS-Anwendungen sind Clients, die mit PTYPE= UPIC-R, APPLI oder SOCKET generiert wurden.

Bei UPIC-Clients wird die Verbindung durch den Client und bei TS-Anwendungen durch den Client oder durch UTM aufgebaut, wobei der Verbindungsaufbau durch UTM nur möglich ist, wenn die TS-Anwendung explizit mit einer PTERM-Anweisung generiert ist.

Falls der Client die Verbindung aufbaut, muss dieser den Namen der UTM-Anwendung sowie Rechnernamen und/oder Rechneradresse kennen. Diese Daten werden auf dem Client-Rechner konfiguriert, z.B. im Transport Name Server (TNS) von CMX/PCMX.

Nach dem erfolgreichen Verbindungsaufbau wird ein UPIC-Client oder eine TS-Anwendung in zwei Schritten angemeldet:

1. Implizites Anmelden über eine **Verbindungs-Benutzerkennung**

   Eine Verbindungs-Benutzerkennung ist einem LTERM-Partner fest zugeordnet und wird für jede TS-Anwendung und jeden UPIC-Client bei der Generierung explizit oder implizit erzeugt:

   – Explizit durch die Angabe bei USER= in der LTERM-Anweisung.
   Für eine so definierte Verbindungs-Benutzerkennung können mit der KDCDEF-Anweisung USER weitere Eigenschaften festgelegt werden.

   – Implizit durch *open*UTM, wenn in der LTERM-Anweisung kein USER angegeben wurde oder wenn es sich um einen LTERM-Pool handelt (TPOOL-Anweisung). Als Verbindungs-Benutzerkennung wird dann der LTERM-Name genommen; bei einem LTERM-Pool setzt sich der LTERM-Name aus dem generierten Präfix und eine laufenden Nummer zusammen, z.B. UPIC0025. Für LTERM-Pools können der Verbindungs-Benutzerkennung mit TPOOL ...USER-KSET= spezielle Keycodes zugeordnet werden. Damit lassen sich die Zugriffsmöglichkeiten der Verbindungs-Benutzerkennung einschränken.

Folgt keine Anmeldung unter einer echten Benutzerkennung, wird die vorläufige Anmeldung der Verbindungs-Benutzerkennung zu einer endgültigen. Dies wird mit einer Meldung protokolliert. Bei UPIC-Clients wird diese Meldung auch ausgegeben, wenn sich dieser anschließend unter einer echten Benutzerkennung anmeldet.

2. Explizites Anmelden über eine **echte Benutzerkennung** (optional)

   Das Verhalten von UPIC-Clients und TS-Anwendungen ist dabei unterschiedlich:

   – Bei UPIC-Clients müssen die Benutzerkennung und die Berechtigungsdaten in den jeweiligen UPIC-Schnittstellenaufrufen gesetzt werden, UPIC übergibt diese Werte anschießend an *open*UTM. *open*UTM führt dann die Anmeldung für die übergebene Benutzerkennung durch. Diese ersetzt die Verbindungs-Benutzerkennung für die Dauer der Conversation. Am Ende der Conversation wird der Benutzer wieder abgemeldet.

     Übergibt der UPIC-Client in den UPIC-Schnittstellenaufrufen keine Berechtigungsdaten, dann ist das Anmelden über eine echte Benutzerkennung nur mit einem entsprechenden Anmeldevorgang möglich, siehe Seite 125.

   – Über eine Transportsystemverbindung kann sich ein Benutzer unter einer echten Benutzerkennung nur dann anmelden, wenn für die Anwendung ein entsprechender Anmeldevorgang generiert ist, siehe Seite 125. Eine Anmeldung mit einer echten Benutzerkennung über den Standard-Anmeldedialog ist nicht möglich.

     Wurde eine TS-Anwendung über eine echte Benutzerkennung angemeldet, dann ersetzt diese Benutzerkennung die Verbindungs-Benutzerkennung für die gesamte Dauer der Verbindung.

   Sowohl für UPIC-Clients als auch für TS-Anwendungen bleibt die Verbindungs-Benutzerkennung mindestens so lange angemeldet wie die echte Benutzerkennung. Tritt ein Verbindungsverlust auf, kann das dazu führen, dass ein erneuter Verbindungsaufbau abgelehnt wird, wenn unter der echten Benutzerkennung noch ein Programm abläuft und deshalb auch die Verbindungs-Benutzerkennung als angemeldet gilt. In diesem Fall muss der Benutzer mit einer Neuanmeldung warten, bis das Programm beendet ist.

8.1.3 Anmeldeverfahren für OSI-TP-Clients

Damit sich ein OSI-TP-Client an die UTM-Anwendung anmelden kann, muss der Client den Namen der UTM-Anwendung, sowie Rechnernamen und/oder Rechneradresse kennen. Diese Daten werden auf dem Client-Rechner konfiguriert.

Bei OSI-TP-Clients kann die Initiative zum Verbindungsaufbau sowohl vom Client als auch von *open*UTM kommen. Dabei können über eine logische Verbindung mehrere parallele Verbindungen aufgebaut werden, die Associations genannt werden. Jeder Association ist ein Association-Name zugeordnet.

Nach erfolgreichem Verbindungsaufbau wird der Client zunächst unter seinem Association-Namen angemeldet. Dieser wird zusammengesetzt aus dem in OSI-LPAP ...,ASSOCIATION-NAMES= angegebenen Namen sowie der laufenden Nummer, z.B. ASSOC03.

Wenn der Client in den jeweiligen Protokollfeldern eine **echte** Benutzerkennung und Berechtigungsdaten übergibt, dann führt *open*UTM anschließend die Anmeldung für die übergebene Benutzerkennung durch. Voraussetzung dafür ist, dass in der OSI-LPAP-Anweisung der entsprechende APPLICATION-CONTEXT ist generiert. Wird keine echte Benutzerkennung übergeben, dann bleibt der Client unter seinem Association-Namen angemeldet.

Am Ende der Conversation wird der Benutzer wieder abgemeldet.

8.1.4 Anmeldeverfahren im World Wide Web über Web*Transactions*

Eine UTM-Anwendung kann über Web*Transactions* an das World Wide Web angeschlossen werden. Damit kann ein Benutzer über einen Browser auf die Services einer UTM-Anwendung zugreifen.

Die Anmeldung über Web*Transactions* ist einfach; sie kann durch die Web*Transactions*-Anwendung gestaltet werden:

1. Der Benutzer gibt in seinem Browser die URL der Web*Transactions*-Anwendung an. Danach wird die Verbindung zur UTM-Anwendung aufgebaut. Eventuell führt die Web*Transactions*-Anwendung zuvor einen eigenen Zwischendialog, z.B. für einen Berechtigungsnachweis für den Zugang zur Web*Transactions*-Anwendung.
2. Der Benutzer muss eventuell wie beim Terminal die UTM-Benutzerkennung und ggf. das Passwort angeben. Ob der Benutzer einen derartigen Berechtigungsdialog führen muss und wie dieser aussieht, hängt jedoch von der Gestaltung der Web*Transactions*-Anwendung ab. Es ist z.B. möglich, die UTM-Benutzerkennung/-Passwort in der Web*Transactions*-Anwendung zu „verstecken“, so dass der Berechtigungsdialog intern abläuft und der Benutzer nach der Eingabe der URL sofort angemeldet ist.

Anschließend kann der Benutzer die Services der Anwendung aufrufen, siehe Seite 133f.

Web*Transactions* nutzt für den Anschluss an die UTM-Anwendung entweder die Terminal-Schnittstelle oder die UPIC-Schnittstelle von UTM. Nähere Informationen finden Sie in den Web*Transactions*-Handbüchern „Anschluss an *open*UTM-Anwendungen über UPIC“ und „Anschluss an OSD-Anwendungen“, falls die Terminal-Schnittstelle verwendet wird.

8.1.5 Mehrfach-Anmeldungen unter einer Benutzerkennung

Ist die Benutzerkennung bei der KDCDEF-Generierung mit RESTART=NO und dem Standardwert MULTI-SIGNON=YES erzeugt worden, dann kann ein Benutzer über verschiedene Verbindungen mehrfach bei *open*UTM angemeldet sein, allerdings nur einmal über eine Verbindung zum Terminal. Mehrfach-Anmeldungen sind nur für echte Benutzerkennungen möglich, **nicht** jedoch für Verbindungs-Benutzerkennungen. Näheres zu Verbindungs-Benutzerkennungen finden Sie auf Seite 120.

Der aktuelle Vorgangskontext belegt in diesem Falle die Ressourcen der Verbindungs-Benutzerkennung (UPIC, TS-Anwendung) bzw. der Association (OSI-TP-Partner).

Meldet sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung über einen OSI TP-Partner an, für dessen Conversation die Functional Unit „Commit“ ausgewählt ist, ist unter dieser Benutzerkennung ebenfalls eine weitere Anmeldung möglich, weil *open*UTM in diesem Falle keinen Vorgangswiederanlauf durchführt und die Benutzerkennung dann so behandelt, als sei kein Wiederanlauf generiert.
Das gleiche gilt, wenn sich der Benutzer über einen OSI-TP-Partner anmeldet und einen asynchronen Auftrag ausführt.

Ansonsten kann sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung zu einem Zeitpunkt nur einmal anmelden, da die für den Vorgangswiederanlauf benötigten Ressourcen der Benutzerkennung zugeordnet werden.

Verhindern von Mehrfach-Anmeldungen für Benutzerkennungen mit RESTART=NO

Über den Parameter MULTI-SIGNON der SIGNON-Anweisung können Sie bei der Generierung festlegen, dass unabhängig von der Wiederanlaufeigenschaft ein Benutzer zu jedem Zeitpunkt nur einmal bei *open*UTM angemeldet sein darf.
Diese Festlegung gilt jedoch nicht für Anmeldungen über einen OSI TP-Partner zur Ausführung von Asynchronaufträgen.

8.1.6 Anmeldeverfahren mit Anmeldevorgängen

Anmeldevorgänge, auch als Event-Services SIGNON bezeichnet, sind selbst programmierte Vorgänge, mit deren Hilfe eigene Anmeldeverfahren definiert werden können. Anmeldevorgänge können von Terminals, UPIC-Clients und TS-Anwendungen genutzt werden, d.h. von Clients, die über PTERM- oder TPOOL-Anweisung generiert sind.

Aufruf von Anmeldevorgängen

Ein Anmeldevorgang ist an den Anwendungsnamen gebunden. Meldet sich ein Client unter einem bestimmten Anwendungsnamen an, dann wird der zu diesem Anwendungsnamen gehörige Anmeldevorgang gestartet und ersetzt das in den vorigen Abschnitten geschilderten Standard-Anmeldeverfahren. Falls mit BCAMAPPL-Anweisungen mehrere Anwendungsnamen generiert sind, dann können mehrere unterschiedliche Anmeldevorgänge in einer Anwendung existieren. Damit lassen sich client-spezifische Anmeldevorgänge erstellen, z.B. einer für Terminals, einer für UPIC-Clients und einer für TS-Anwendungen. Weitere Details finden Sie in Abschnitten „Anmeldevorgänge für Terminals“ auf Seite 126 bis „Anmeldevorgang für UPIC-Clients“ auf Seite 127.

Ist für einen Anwendungsnamen kein Anmeldevorgang generiert, dann durchläuft der Client das Standard-Anmeldeverfahren.

Generieren von Anmeldevorgängen

Anmeldevorgänge werden wie folgt generiert, siehe auch *open*UTM-Handbuch „Anwendungen generieren“:

- Mit TAC KDCSGNTC wird der Anmeldevorgang für den Standard-Anwendungsnamen (definiert in MAX APPLINAME) generiert.
- Mit BCAMAPPL *appliname2...*,SIGNON=*signon-tac* wird der Anmeldevorgang für den Anwendungsnamen *appliname2* generiert. *signon-tac* muss in einer TAC-Anweisung definiert sein.
- Sollen Anmeldevorgänge auch von UPIC-Clients genutzt werden können, dann muss zusätzlich SIGNON ...,UPIC=YES generiert werden.

Für jeden dieser TACs ist auch eine PROGRAM-Anweisung nötig. Dort wird der Name des Teilprogramms angegeben, das im Anmeldevorgang als erstes durchlaufen wird.

Programmieren von Anmeldevorgängen

Für die Programmierung eines Anmeldevorgangs gibt es die speziellen KDCS-Aufrufe SIGN ST, SIGN ON und PEND PS. Wie ein Anmeldevorgang programmiert werden kann und welche Regeln dabei zu beachten sind, ist ausführlich im entsprechenden Abschnitt im *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

8.1.6.1 Anmeldevorgänge für Terminals

Ein Anmeldevorgang für Terminals setzt sich im Allgemeinen aus zwei Teilen zusammen:

Anmeldevorgang	
Teil 1	Teil 2
Der Vorgang fordert den Benutzer auf, sich zu legitimieren, liest mit MGET die Berechtigungsdaten und übergibt diese zur Prüfung an *open*UTM. Der Vorgang ist noch **keiner** Benutzerkennung zugeordnet.	*open*UTM hat die Berechtigungsdaten akzeptiert und hat den Anmeldevorgang der ermittelten Benutzerkennung zugeordnet.

Zwischen dem ersten und zweiten Teil des Anmeldevorgangs kann *open*UTM ggf. einen Zwischendialog einfügen, der über *open*UTM-Meldungen gesteuert wird. Dieser Zwischendialog dient dazu, eine Magnetstreifenkarte einzulesen, oder ein mit USER...,PASS=(...,DARK) generiertes Passwort. In diesen Fällen fordert *open*UTM den Benutzer wie beim Standard-Anmeldedialog auf, diese Information einzugeben. Dieser Zwischendialog bietet dem Anwender zusätzliche Sicherheit für den Zugang zu seinen Anwendungen. Im Zwischendialog ist es möglich (wie beim Standard-Anmeldedialog), das bisherige Passwort durch ein neues zu überschreiben.

Spezialfälle des Anmeldevorgangs für Terminals

Bei der Generierung von LTERM-Partnern mit automatischem KDCSIGN und der Anmeldung über Verteiler muss der Anmeldevorgang entsprechend angepasst werden.

LTERM-Partner mit automatischem KDCSIGN

Der Anmeldevorgang erhält beim Aufruf SIGN ST die Information, dass die Benutzerkennung bereits bekannt ist. Abhängig von der Generierung kann jetzt ein Zwischendialog zum Anfordern einer Magnetstreifenkarte oder eines Passworts durchgeführt werden.

Anmeldung über Verteiler

Wenn der Verteiler selbst ein Anmeldeverfahren durchführt, leitet er die Berechtigungsdaten über das Protokoll PUTMMUX zur Prüfung an *open*UTM weiter. Sind die Daten korrekt, startet *open*UTM den Anmeldevorgang, der mit dem Aufruf SIGN ST die entsprechende Information erhält. Sind die Daten nicht korrekt, wird der Anmeldevorgang nicht gestartet, und der Verteiler muss seinerseits den Benutzer am Terminal informieren.

8.1.6.2 Anmeldevorgang für TS-Anwendungen

Beim Start des Anmeldevorgangs ist der Benutzer unter der Verbindungs-Benutzerkennung vorläufig angemeldet.

Im Anmeldevorgang können über den Aufruf SIGN ON die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn *open*UTM die Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmeldevorgangs unter der angegebenen Benutzerkennung angemeldet. Die Anmeldung wird abgelehnt, wenn die Berechtigungsdaten der TS-Anwendung nicht korrekt sind oder wenn unter der Verbindungs-Benutzerkennung ein Vorgang offen ist.

Ist die Anmeldung unter einer echten Benutzerkennung einmal fehlgeschlagen, dann muss in demselben Anmeldevorgang eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmeldevorgangs die Verbindung abgebaut. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Wird im Anmeldevorgang keine Benutzerkennung übergeben, dann wird der Benutzer beim ordnungsgemäßen Abschluss des Anmeldevorgangs endgültig unter der Verbindungs-Benutzerkennung angemeldet.

8.1.6.3 Anmeldevorgang für UPIC-Clients

Beim Anmelden über einen Anmeldevorgang sind zwei Fälle zu unterscheiden:

- Der UPIC-Client übergibt im UPIC-Protokoll Berechtigungsdaten an *open*UTM. Wenn *open*UTM die Daten akzeptiert, dann wird der Anmeldevorgang unter der übergebenen echten Benutzerkennung gestartet und der Client wird beim ordnungsgemäßen Abschluss des Anmeldevorgangs unter dieser Benutzerkennung angemeldet.
- Falls UPIC-Client im UPIC-Protokoll keine Berechtigungsdaten übergibt, dann wird der Anmeldevorgang unter der Verbindungs-Benutzerkennung gestartet. Im Anmeldevorgang können die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn *open*UTM diese Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmeldevorgangs unter dieser Benutzerkennung angemeldet. Werden keine Berechtigungsdaten übergeben, läuft die Conversation unter der Verbindungs-Benutzerkennung.

Ist die Anmeldung unter einer echten Benutzerkennung fehlgeschlagen, dann muss eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmeldevorgangs die Conversation beendet. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Damit Client-Programme unabhängig davon eingesetzt werden können, ob die UTM-Anwendung einen Anmeldevorgang verwendet oder nicht, können nicht gelesene Nachrichten vom Client bei Beendigung eines Teilprogramms des Anmeldevorgangs mit PEND PA, PEND PR, PEND PS oder PEND FC ohne vorhergehenden MPUT im Folgeteilprogramm gelesen werden.

8.1.6.4 Anwendungsmöglichkeiten für Anmeldevorgänge

Anmeldevorgänge bieten dem Anwender eine Reihe praktischer Nutzungsmöglichkeiten, die im Folgenden skizziert werden:

- Das Anmeldeverfahren für Terminals kann als formatierter Dialog ablaufen.
- Startformate können aktuelle Daten enthalten, wie Datum, Uhrzeit, Bulletin etc.
- In einer Anwendung mit mehrsprachigen Benutzern kann man LTERM-spezifische Startformate verwenden, um einen Begrüßungsbildschirm in der jeweiligen Landessprache auszugeben. Diese Möglichkeit gibt es nur bei Terminals.
- Userspezifische Startformate sind zweckmäßig, um Informationen zu zeigen, die spezifisch für eine Benutzergruppe sind (Bulletin, Menü etc.).
- TS-Anwendungen können sich über einen Anmeldevorgang mit einer echten Benutzerkennung an eine UTM-Anwendung anmelden. Dadurch werden sie in das Zugangs- und Zugriffskonzept von *open*UTM eingebunden.
- Der vom Benutzer eingegebene Name kann in eine Benutzerkennung umgesetzt werden, die per Generierung definiert ist (USER *username*).
- Im Falle eines DB/DC-übergreifenden Berechtigungskonzeptes kann man im 2. Teil des Anmeldevorgangs mit einem Datenbankaufruf das aktuelle Berechtigungsprofil für diesen USER aus der Datenbank holen und evtl. in einem benutzerspezifischen Langzeitspeicher (ULS) speichern.
- Der Anmeldevorgang kann den Benutzer im 2. Teil zur Änderung seines Passwortes auffordern, z.B. weil die Zeitspanne überwacht wird, in welcher der Benutzer dasselbe Passwort verwendet.
- Es kann eine Statistik über alle versuchten und erfolgreichen Anmeldungen erstellt werden.
- Das Formatsteuerungssystem FHS bietet die Möglichkeit, für Terminals die P-Tasten über ein Globalattribut eines #-Formats zu laden. Der Anmeldevorgang kann im 2. Teil die P-Tasten beispielsweise benutzerspezifisch laden.
- Auch im Falle eines nachfolgenden Vorgangswiederanlaufs kann der Anmeldevorgang dem Benutzer nützliche Informationen geben. Hierzu zählen Bulletin, Anzeige der Tastaturbelegung oder Anzeige des Vorgangswiederanlaufs. Dies erfordert natürlich einen zusätzlichen Dialogschritt.

- Wenn *open*UTM nach einem Aufruf SIGN OB (= KDCOFF BUT per Programm) den Anmeldevorgang startet, kann es sinnvoll sein, dass mit MGET die letzte Eingabe vom Terminal gelesen wird, wenn dort bereits neue Berechtigungsdaten eingetragen wurden.

8.1.6.5 Eigenschaften von Anmeldevorgängen

Ausgabe der letzten Dialognachricht durch den Anmeldevorgang

Liegt kein Vorgangswiederanlauf vor und wird der Anmeldevorgang mit MPUT PM und PEND FI beendet, dann wird die letzte Dialognachricht ausgegeben. Der Anwender kann dann nach Abschluss des Anmeldevorgangs mit dem gleichen Bildschirm weiterarbeiten, mit dem die letzte Sitzung beendet wurde - unabhängig davon, ob dies innerhalb oder außerhalb eines Vorgangs geschah.

Meldungen

Verwendet eine UTM-Anwendung einen Anmeldevorgang, werden die folgenden Meldungen nicht erzeugt (und demzufolge auch nicht an SYSLOG und MSGTAC ausgegeben):

K001, K002, K004 bis K008.

Die Meldung K033 (erfolgreiche Anmeldung) wird auch bei Verwendung eines Anmeldevorgangs ausgeben.

Fehlversuche beim Anmeldevorgang

Beim Anmeldevorgang können Fehlversuche des Benutzers beim Anmelden abgefangen werden: Akzeptiert *open*UTM die eingegebenen Berechtigungsdaten des Benutzers **nicht**, dann kann der Anmeldevorgang den Benutzer auffordern, die Eingabe zu wiederholen. Die maximale Anzahl der Eingabeversuche ist programmierbar. Wird sie überschritten, dann sollte sich der Anmeldevorgang beenden. Bei TS-Anwendungen und bei Terminals baut UTM die Verbindung ab, bei UPIC wird nur die Conversation beendet.

*open*UTM zählt außerdem alle Fehlversuche von einem Client mit, die in ununterbrochener Folge auftreten, auch über eine Folge von Anmeldevorgängen hinweg. Nach einer bei der Generierung festzulegenden Anzahl von fehlerhaften Anmeldeversuchen (siehe KDCDEF-Anweisung SIGNON, Operand SILENT-ALARM im *open*UTM-Handbuch „Anwendungen generieren“) meldet *open*UTM dieses Ereignis nach SYSLOG (stiller Alarm, Meldung K094). Anmeldeversuche unberechtigter Personen können aufgedeckt und mit einer MSGTAC-Routine abgewehrt werden.

Fehlverhalten des Anmeldevorgangs

*open*UTM kontrolliert, ob die Regeln für den Anmeldevorgang eingehalten werden. Das bietet auch einen Schutz gegen eventuelle Manipulationen der Teilprogramme des Anmeldevorgangs. Bei Fehlern dieser Art bricht *open*UTM den Anmeldevorgang mit PEND ER ab. Danach wird bei TS-Anwendungen und Terminals die Verbindung abgebaut; bei UPIC wird nur die Conversation beendet.

8.1.6.6 Beispielprogramme für Anmeldevorgang

Zusammen mit *open*UTM werden Teilprogramme als COBOL-Quellprogramme ausgeliefert, die einen fertigen Anmeldevorgang mit formatierter Schnittstelle zum Terminal realisieren. Dieser Anmeldevorgang ist für alle Generierungsvarianten geeignet. Das verwendete Format enthält englische Texte.

Der Anwender kann diese Vorlage nach seinen Wünschen abändern und erhält so auf einfache Weise ein Anmeldeverfahren mit formatierter Schnittstelle zum Benutzer. Er muss dadurch mit der Programmierung nicht völlig neu beginnen.

8.1.7 Verhalten bei gesperrten Clients/LTERM-Partnern

Verhalten bei gesperrten Clients

Clients können per Generierung (PTERM...,STATUS=OFF) oder per Administration gesperrt werden. Das Sperren eines Client hat folgende Wirkungen:

- Der Verbindungswunsch wird abgelehnt.
- Eine bestehende Verbindung bleibt erhalten; die Sperre wird erst dann wirksam, wenn von diesem Client ein neuer Verbindungswunsch kommt.

Verhalten bei gesperrten LTERM-Partnern

LTERM-Partner können per Generierung (LTERM...,STATUS=OFF) oder per Administration gesperrt werden.

Bei UPIC-Clients und TS-Anwendungen wirkt das Sperren des LTERM-Partners wie das Sperren des Clients.

Bei Terminals hat das Sperren eines LTERM-Partners folgende Wirkungen:

- Der Verbindungswunsch wird ausgeführt, aber nach dem Verbindungsaufbau wird die Meldung ausgegeben:

```
K027 LTERM-Partner &LTERM gesperrt - Administrator verstaendigen
     oder KDCOFF eingeben.
```

- Eine bestehende Verbindung bleibt erhalten, die nächste Eingabe vom Terminal wird mit der Meldung K027 quittiert.

8.2 Anmeldeverfahren ohne Benutzerkennungen

Bei UTM-Anwendungen, für die keine Benutzerkennungen generiert sind, nimmt *open*UTM keine Berechtigungsprüfung vor. Die Clients werden unter ihrem LTERM-Namen bzw. Associations-Namen angemeldet. UPIC-Clients und OpenCPIC-Clients dürfen in diesem Fall keine echte Benutzerkennung übergeben.

Verwendet die UTM-Anwendung Anmeldevorgänge (Seite 125f), dann kann damit anschließend eine anwendungseigene Berechtigungsprüfung durchgeführt werden, zum Beispiel anhand einer Datenbank mit Berechtigungsdaten.

Wird kein Anmeldevorgang verwendet, dann kann der Benutzer nach erfolgreichem Verbindungsaufbau zu der UTM-Anwendung sofort mit dieser Anwendung arbeiten. Bei Terminals und TS-Anwendungen erhält der Benutzer von *open*UTM eine Ausgabe, die davon abhängt, ob für diesen LTERM-Partner noch ein offener Vorgang bekannt ist:

- Wenn für den LTERM-Partner in der Anwendung kein offener Vorgang bekannt ist, gibt *open*UTM folgende Meldung aus:

  ```
  K001 Verbunden mit Anwendung beispiel – Bitte Eingabe
  ```

  Bei Terminals wird, falls generiert, das Startformat zu diesem LTERM-Partner ausgegeben. Der Benutzer kann dann Vorgänge starten und UTM-Benutzer-Kommandos eingeben.

- Wenn für diesen LTERM-Partner in der Anwendung noch ein offener Vorgang bekannt ist, so bekommt der Benutzer die Ausgabe vom letzten Sicherungspunkt seines unterbrochenen Vorgangs auf den Bildschirm, und er kann den Vorgang fortsetzen. Sehen Sie dazu auch *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“. Dafür ist unter anderem Voraussetzung, dass für diesen LTERM-Partner RESTART=YES generiert wurde. Das heißt aber auch, dass der Benutzer eventuell den Vorgang eines anderen Benutzers fortsetzen kann.

Beachten Sie, dass *open*UTM in einer Anwendung ohne Benutzerkennungen einen Vorgang an den LTERM-Partner koppelt. Ein unterbrochener Vorgang kann deshalb auch nur vom selben Client fortgesetzt werden. Es sei denn, die Zuordnung LTERM-Partner und physikalischer Client (festgelegt in der PTERM-Anweisung) wird per Administration, z.B. mit dem Administrationskommando KDCSWTCH, geändert.

Das Verhalten bei gesperrten Clients ist das gleiche wie mit Benutzerkennungen, siehe Seite 131.

ACHTUNG!
In einer UTM-Anwendung ohne Benutzerkennungen hat jeder Benutzer die Administrationsberechtigung.

8.3 UTM-Services aufrufen

Ist die UTM-Berechtigungsprüfung erfolgreich verlaufen, so ist der Benutzer berechtigt, mit der UTM-Anwendung zu arbeiten, d.h er kann neue Vorgänge (=Services) starten (siehe unten) oder offene Vorgänge fortsetzen.

Die Abschnitte „Vorgänge vom Terminal aus starten“ auf Seite 133 bis „Vorgänge von TS-Anwendungen aus starten“ auf Seite 134 zeigen, wie bei den einzelnen Client-Typen neue Vorgänge gestartet werden. Der Abschnitt „Vorgangswiederanlauf“ auf Seite 135 beschreibt die Situation, wenn für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist.

8.3.1 Vorgänge vom Terminal aus starten

Nach erfolgreicher Anmeldung kann der Benutzer einen Vorgang starten, indem er einen Transaktionscode (TAC) eingibt oder eine entsprechend generierte Funktionstaste drückt.

Starten eines Vorgangs durch Eingabe eines Transaktionscodes

Wenn kein Anmeldevorgang durchlaufen wird, kann der Benutzer folgende Situationen vorfinden:

- Im Zeilenmodus gibt *open*UTM die folgende Meldung aus:

  ```
  K008 Anmeldung akzeptiert - Bitte Eingabe
  ```

  Der Benutzer kann einen Vorgang starten, indem er einen TAC und eventuell eine Nachricht eingibt. Die ersten acht Zeichen seiner Eingabe werden von *open*UTM als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, dann muss er durch ein Leerzeichen von der Nachricht getrennt sein.

- Wenn für den Benutzer ein Startformat generiert wurde, dann muss er die entsprechenden Felder des Startformats ausfüllen. Das Feld mit dem TAC, auch Steuerfeld genannt, steht dabei nicht notwendigerweise an erster Stelle.

Wenn ein Anmeldevorgang durchlaufen wird, dann bestimmt der Anmeldevorgang den nächsten Schritt. Der Benutzer erhält dann eine Ausgabe im Format- oder Zeilenmodus oder es wird direkt ein Vorgang gestartet.

Starten eines Vorgangs über eine Funktionstaste

Ein TAC kann per Generierung auf eine Funktionstaste gelegt werden, z.B. F10, siehe KDCDEF-Anweisung SFUNC. Wenn der Benutzer diese Taste betätigt, dann wird der Vorgang gestartet, unabhängig davon, ob ein Benutzer im Zeilenmodus arbeitet oder ein Startformat vor sich hat.

Eingabe ungültiger Transaktionscodes

Wenn der Benutzer einen falschen TAC eingibt, dann erhält er die Meldung

```
K009  Der Transactionscode <tac> ist ungültig – Bitte Eingabe
```

Falls in der Anwendung ein Dialogvorgang BADTACS generiert ist, dann wird stattdessen der Vorgang BADTACS gestartet. Nachdem der Dialogvorgang BADTACS beendet wurde, bleibt der Benutzer angemeldet und kann wie oben beschrieben einen Vorgang starten.

8.3.2 Vorgänge vom UPIC-Client und OSI-TP-Client aus starten

Nach dem Verbindungsaufbau können die OSI-TP- oder UPIC-Clients Conversations starten. Dazu wird der TAC durch den Client gesetzt, z.B. über die Funktion *Set_TP_Name* der CPI-C-Schnittstelle oder durch einen entsprechenden Eintrag in der Side Information Datei. Dieser TAC wird an *open*UTM übergeben, eventuell zusammen mit Berechtigungsdaten. Nach erfolgreicher Berechtigungsprüfung gilt:

- Bei OpenCPIC-Clients und bei UPIC-Clients ohne Anmeldevorgang wird der zum TAC gehörige Vorgang sofort gestartet.
- Bei UPIC-Clients mit Anmeldevorgang wird der zum TAC gehörige Vorgang erst nach dem Ende des Anmeldevorgangs gestartet.

Am Ende der Conversation wird der Benutzer wieder abgemeldet, wenn er sich für diese Conversation unter einer echten Benutzerkennung angemeldet hat.

8.3.3 Vorgänge von TS-Anwendungen aus starten

TS-Anwendungen verhalten sich ähnlich wie Terminals:

- Ohne Anmeldevorgang erhält die TS-Anwendung die Meldung K001, falls dieser Meldung das Meldungsziel PARTNER zugewiesen wurde, siehe Beschreibung des Tools KDCMMOD im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.

  Anschließend kann die TS-Anwendungen einen Vorgang starten, indem sie einen TAC und eventuell eine Nachricht an die UTM-Anwendung übergibt. Dazu werden die ersten 8 Zeichen der Nachricht als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, muss er durch Leerzeichen von der Nachricht getrennt sein.
- Wird ein Anmeldevorgang durchlaufen, dann bestimmt dieser den nächsten Schritt. Der Anmeldevorgang kann entweder direkt einen Vorgang starten oder eine Ausgabenachricht an die TS-Anwendung schicken. Wird eine Ausgabenachricht an die TS-Anwendung geschickt, dann muss die nächste Nachricht in den ersten 8 Zeichen einen TAC enthalten, d.h. es gilt das gleiche wie ohne Anmeldevorgang, siehe oben.

Nach dem Beenden des Vorgangs wird die TS-Anwendung von UTM abgemeldet.

8.3.4 Vorgangswiederanlauf

Wenn sich ein Client unter einer bestimmten Benutzerkennung anmeldet, die mit RESTART=YES generiert ist, und wenn für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist, dann wird in der Regel ein Vorgangswiederanlauf durchgeführt. Beim Vorgangswiederanlauf sendet *open*UTM die Ausgabe vom letzten Sicherungspunkt des unterbrochenen Vorgangs an den Client, der Benutzer kann den Vorgang anschließend fortsetzen.

Abhängig von der Art des Client und vom Anmeldeverfahren gilt für den Vorgangswiederanlauf Folgendes:

- Standard-Anmeldeverfahren für Terminals und TS-Anwendungen:
  *open*UTM führt den Vorgangswiederanlauf automatisch durch.
- Standard-Anmeldeverfahren für UPIC-Clients und OSI-TP-Clients:
  Der Client muss eine spezielle Conversation starten, die mittels des UTM-Benutzerkommandos KDCDISP den Wiederanlauf anfordert (siehe z.B. Handbuch „*open*UTM-Client fürTrägersystem UPIC“). Über OSI- TP-Clients ist kein Fortsetzen des Vorgangs möglich, wenn die Functional Unit „Commit“ ausgewählt wurde.
- Anmelden über Anmeldevorgang:
  Der Anmeldevorgang muss den Wiederanlauf initiieren oder den offenen Vorgang abnormal beenden.



In einer Anwendung mit Benutzerkennungen ist ein Vorgang an die Benutzerkennung gebunden. Deshalb kann der Benutzer einen unterbrochenen Vorgang auch an einem anderen Client fortsetzen, sofern der LTERM-Partner des Client die Berechtigung dazu hat und der Typ des Clients gleich bleibt.

8.4 Berechtigungskonzept von *open*UTM

*open*UTM bietet zusätzlich zum Zugangsschutz durch Benutzerkennungen ein ausgefeiltes Zugangs- und Zugriffskonzept. Damit lässt sich steuern, welcher Benutzer über welche LTERM-Partner auf welche Services der UTM-Anwendung zugreifen darf.

Dabei gibt es die benutzerorientierte Variante (**Lock-/Keycode**-Konzept) und die rollenorientierte Variante (**Access-List-**Konzept). Diese Varianten werden mit Hilfe von Lockcodes, Access-Lists, Keysets und Keycodes generiert:

- Ein Service wird entweder mit Lockcodes (Lock-/Keycode-Konzept) oder mit einer Access-List (Access-List-Konzept) geschützt (TAC-Anweisung, LOCK= bzw. ACCESS-LIST=).
- Eine Benutzerkennung erhält ein Keyset mit einem oder mehreren Keycodes (USER-Anweisung, KSET=). Die Keycodes definieren die Berechtigungen.
- Ein LTERM-Partner erhält ein Keysets mit einem oder mehreren Keycodes sowie Lockcodes, wenn das Lock-/Keycode-Konzept verwendet wird (LTERM- bzw. TPOOL-Anweisung, Operanden KSET= und LOCK=).
- Keysets werden eigens in KSET-Anweisungen definiert

Die folgende Tabelle zeigt für die beiden Konzept-Varianten, unter welchen Voraussetzungen ein Benutzer sich anmelden kann und wann er einen Vorgang starten darf oder fortsetzen darf (nach einem Vorgangswiederanlauf).

Tätigkeit	**Voraussetzungen**	
	Lock-/Keycode-Konzept	**Access-List-Konzept**
Anmelden über bestimmten LTERM-Partner	Ein Keycode der Benutzerkennung stimmt mit dem Lockcode des LTERM-Partners überein.	Anmelden immer möglich.
Starten eines Vorgangs	Benutzerkennung und LTERM-Partner besitzen einen Keycode, der mit dem Lockcode des TACs übereinstimmt.	Benutzerkennung und LTERM-Partner besitzen jeweils einen Keycode, der in der Access-List des TACs enthalten ist. Die Keycodes von Benutzerkennung und LTERM müssen nicht identisch sein.
Vorgang fortsetzen (nach Vorgangswiederanlauf)	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss mit dem Lockcode des Folge-TACs übereinstimmen.	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss in der Access-List des Folge-TACs enthalten sein.

Meldungen bei fehlenden Berechtigungen

Wenn Berechtigungen fehlen, kann der Terminalbenutzer folgende Meldungen erhalten (beim Anmeldevorgang kommt ein entsprechender Returncode):

```
K005 Die Benutzerkennung user ist gesperrt
```

wenn der Keycode des Benutzers nicht mit dem Keycode des LTERM-Partners übereinstimmt (Anmeldevorgang: Returncode U02).

```
K009 Der Transaktionscode ist ungueltig
```

wenn der Benutzer oder das LTERM nicht die Berechtigung haben, den Vorgang zu starten. Wenn ein BADTAC-Vorgang generiert ist, dann wird stattdessen der BADTAC-Vorgang gestartet.

```
K123 LTERM hat nicht die Berechtigung den Vorgang fortzusetzen - Bitte KDCSIGN
```

wenn der LTERM-Partner, über den sich der Benutzer beim Vorgangswiederanlauf angemeldet hat, nicht die Berechtigung hat, den Folge-TAC zu starten (Anmeldevorgang: Returncode U16). Diese Meldung kann insbesondere dann auftreten, wenn ein Benutzer den Vorgang von einem anderen Terminal und damit einem anderen LTERM fortsetzt.



Weitere Informationen finden Sie im *open*UTM-Handbuch „Konzepte und Funktionen” sowie im *open*UTM-Handbuch „Anwendungen generieren“.

8.5 Abmelden von der UTM-Anwendung

Die folgenden Abschnitte beschreiben die verschiedenen Möglichkeiten, wie sich ein Client bei der UTM-Anwendung abmelden kann oder ein Client durch UTM abgemeldet wird. Dabei unterscheiden sich Terminals von allen anderen Clients, denn nur von Terminals aus können sich Benutzer explizit von der Anwendung abmelden.

Abmelden bei Zeitüberschreitung - Zeitüberwachung

Bei der Generierung können maximale Wartezeiten festegelegt werden durch:

- die Operanden TERMWAIT= (PEND KP-Timer) und PGWTTIME= (PGWT-Timer) in der KDCDEF-Steueranweisung MAX
- den Operanden IDLETIME= (Transaktionsende-Timer) der PTERM-Anweisung bzw. der OSI-LPAP-Anweisung (bei OSI-TP-Clients).

Wenn eine mit diesen Timern eingestellte Wartezeit abgelaufen ist, dann wird bei Terminals folgende Meldung ausgegeben:

```
K021 Eine Eingabe ist nicht in der vorgegebenen Zeit erfolgt
```

Anschließend meldet *open*UTM die Benutzerkennung ab und baut die Verbindung zum Client ab. Der Client kann sich danach wieder bei der Anwendung anmelden und den Vorgang fortsetzen, siehe Abschnitt „Vorgangswiederanlauf“ auf Seite 135.

Abmelden mit KDCOFF-Kommando

Der Terminal-Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF bzw. KDCOFF BUT von der UTM-Anwendung abmelden. Sehen Sie hierzu auch das UTM-Benutzerkommando KDCOFF auf Seite 146.

KDCOFF aus einem Programm

*open*UTM bietet die Funktionsaufrufe SIGN OF und SIGN OB, mit denen in einem Dialog-Teilprogramm die Wirkung des Benutzerkommandos KDCOFF bzw. KDCOFF BUT ausgelöst werden kann. SIGN OF/OB ist für Terminals, UPIC-Clients und TS-Anwendungen möglich. In Teilprogrammen, die für einen OSI-TP-Client laufen, sind diese Aufrufe nicht erlaubt.

SIGN OF und SIGN OB wirken wie folgt:

SIGN-Aufruf	Kommando	Auswirkung
SIGN OF	KDCOFF	*open*UTM baut Verbindung zum Client ab.
SIGN OB	KDCOFF BUT	Für Terminals bleibt die Verbindung bleibt erhalten, der Benutzer wird abgemeldet. Bei UPIC-Clients und TS-Anwendungen wird die Verbindung abgebaut (wie SIGN OF).

Der Aufruf wirkt bei Terminals und UPIC-Clients/TS-Anwendungen unterschiedlich:

- Bei Terminals gibt *open*UTM zunächst die MPUT-Nachricht und die Meldung K095 an das Terminal aus. Erst durch die nächste (beliebige) Eingabe vom Terminal wird der Benutzer abgemeldet und (bei SIGN OF) die Verbindung abgebaut.
- Bei UPIC-Clients und TS-Anwendungen wird die MPUT-Nachricht gesendet und anschließend sofort der Verbindungsabbau initiiert.

Im Folgenden werden einige Anwendungsmöglichkeiten für den Funktionsaufruf SIGN OF/OB skizziert:

- Anwendungen mit besonderen Sicherheitsanforderungen. Nach dem Anmelden darf ein Benutzer nur einen einzigen Vorgang bearbeiten.
- Verbesserter Datenschutz: Der Bildschirm kann durch den letzten MPUT überschrieben werden, es bleiben keine vorgangsspezifischen Daten am Bildschirm stehen.
- Der Steuerteil des Bildschirms bietet als mögliche nächste Aktionen auch „Abmelden“ oder „Neu-Anmeldung“ an. Das Folgeteilprogramm erzeugt dann abhängig von der Eingabe einen Aufruf SIGN OF bzw. SIGN OB. Nach der Dialogausgabe dieses Teilprogramms und der darauf folgenden Eingabe wird entweder die Verbindung zum Terminal abgebaut oder aber der Anmeldevorgang gestartet.

8.6 UTM-Benutzerkommandos für Terminals

Dieser Abschnitt enthält eine Beschreibung aller UTM-Benutzerkommandos, die dem Terminal-Benutzer nach dem Anmelden zur Verfügung stehen (das zum Anmelden benötigte Kommando KDCSIGN ist auf Seite 115 beschrieben):

- KDCFOR, um das Basisformat auszugeben
- KDCOUT, um asynchrone Nachrichten anzufordern
- KDCDISP, um die letzte Dialognachricht nochmals anzufordern
- KDCLAST, um die letzte Ausgabe zu wiederholen
- KDCOFF, um sich abzumelden

UTM-Benutzerkommandos können bei der Generierung der Anwendung mit der KDCDEF-Anweisung SFUNC einer K-/F-Taste zugeordnet werden, und können dann über Funktionstaste eingegeben werden.

KDCFOR - Basisformat ausgeben

Mit dem KDCFOR-Kommando kann der Benutzer ein Basisformat ausgeben. Ein Basisformat dient dazu, die Daten für einen Auftrag am Bildschirm in Formatfelder einzutragen.

Durch Eingabe von KDCFOR { *formatkennzeichen*[1] | '*formatkennzeichen*'[2] } wird das Format mit dem angegebenen Formatkennzeichen am Terminal ausgegeben. Dieses Format gilt solange als Basisformat des Benutzers, bis ein weiteres KDCFOR-Kommando mit einem anderen Formatkennzeichen eingegeben wird.

Mit dem Kommando KDCFOR ohne *formatkennzeichen* kann der Benutzer das gerade geltende Basisformat am Terminal ausgeben.



Mit dem KDCFOR-Kommando kann man **kein** #Format ausgeben.

Im Fehlerfall gibt *open*UTM eine der folgenden Meldungen aus:

`K013 Fehlerhafte Eingabe im Kommando - Bitte Eingabe`

Das KDCFOR-Kommando hat nicht die vorgeschriebene Form.

`K014 kein Basisformat verfuegbar - Bitte Eingabe`

Der Benutzer hat ein KDCFOR-Kommando ohne Formatkennzeichen angegeben, es ist aber noch kein Basisformat bekannt.

`K015 Formatierungsfehler ..... - Bitte Eingabe`

Beim Versuch, das Basisformat auszugeben, trat ein Fehler auf.

`K003 Kommando KDCFOR ist jetzt nicht erlaubt.`

Das KDCFOR-Kommando ist auf dieser Stufe des Dialogs (mit der UTM-Anwendung) nicht erlaubt.

[1] wenn Präfix für *formatkennzeichen* = *

[2] wenn Präfix für *formatkennzeichen* = *, +, oder -

KDCOUT - Asynchrone Nachricht ausgeben

Mit dem KDCOUT-Kommando kann der Benutzer die Ausgabe asynchroner Nachrichten anfordern. Bei der Generierung wird mit dem Operanden ANNOAMSG={ YES | NO } in der LTERM bzw. TPOOL-Anweisung festgelegt, wie *open*UTM asynchrone Nachrichten an dieses Terminal (LTERM) bzw. diesem LTERM-Pool (TPOOL) ausgeben soll.

Ausgabe bei ANNOAMSG=Y

In diesem Fall, kündigt *open*UTM asynchrone Nachrichten mit der folgenden Meldung an:

```
K012 nnn Nachricht(en) vorhanden
```

Die Meldung erscheint zusammen mit der nächsten Dialogausgabe an dieses Terminal in der Systemzeile. Mit *nnn* wird die Anzahl der asynchronen Nachrichten angegeben. Der Benutzer kann diese Nachrichten mit dem Kommando KDCOUT abholen. Sind jedoch keine Nachrichten für das Terminal vorhanden, so meldet *open*UTM:

```
K020 keine Nachricht vorhanden
```

Wird eine asynchrone Nachricht mit KDCOUT abgeholt, dann wird sie durch die nächste Eingabe gelöscht, außer bei Eingabe von KDCLAST (siehe Seite 145).

Wenn sich der Benutzer in einem formatgesteuerten Dialog mit einem Vorgang befindet und mit KDCOUT eine asynchrone Nachricht ausgibt, so zerstört er damit das letzte Ausgabeformat. Zur Fortsetzung des Vorgangs muss er das Ausgabeformat mit dem Benutzerkommando KDCDISP (siehe Seite 144) wiederherstellen. Tut er das nicht, so führt *open*UTM selbst einen KDCDISP durch. Der Benutzer muss anschließend seine Eingabe wiederholen.

Die KDCDEF-Anweisung LTERM ..., RESTART= NO hat zur Folge, dass beim Verbindungsaufbau oder -abbau zu diesem LTERM-Partner anstehende asynchrone Nachrichten gelöscht werden.

Die Funktionsvarianten von *open*UTM haben folgende Auswirkungen auf die Behandlung von asynchronen Nachrichten:

- Bei UTM-S-Anwendungen bleiben asynchrone Nachrichten auch über Unterbrechungen des Anwendungslaufes hinweg gesichert, bis sie mit KDCOUT abgeholt werden.
- Bei UTM-F-Anwendungen werden asynchrone Nachrichten nur während des Anwendungslaufes gespeichert. Sie gehen mit der Beendigung des Anwendungslaufes verloren.

Ausgabe bei ANNOAMSG=N

Wenn ein Client mit einem LTERM-Partner verbunden ist, der mit LTERM...,ANNOAMSG=N oder TPOOL ...,ANNOAMSG=N generiert ist, dann ist an diesem Client das Kommando KDCOUT nicht erlaubt. ANNOAMSG=N bedeutet, dass *open*UTM asynchrone Nachrichten sofort, d.h. ohne vorherige Ankündigung, am Terminal ausgibt. *open*UTM weist deshalb das Kommando KDCOUT an dieses Terminal mit der Meldung K003 ab:

```
K003 Das Kommando KDCOUT ist jetzt nicht erlaubt.
```

In einer UTM-Anwendung mit Benutzerkennungen werden asynchrone Nachrichten frühestens nach der erfolgreich abgeschlossenen Berechtigungsprüfung ausgegeben.

Möchte der Benutzer nach Ausgabe der asynchronen Nachricht seinen Dialog mit der Anwendung fortsetzen, so muss er zuvor mit KDCDISP den Bildschirm der letzten Dialogausgabe anfordern. Andernfalls führt *open*UTM einen automatischen KDCDISP durch. Dieser automatische Bildschirmwiederanlauf wird von *open*UTM nur dann unterdrückt, wenn die asynchrone Nachricht mit demselben Format ausgegeben wurde wie die letzte Dialogausgabe.



Bei #-Formaten erfolgt **immer** ein automatischer Wiederanlauf.

KDCDISP - Letzte Dialognachricht ausgeben

In einer laufenden UTM-Anwendung kann sich der Benutzer mit dem KDCDISP-Kommando die letzte Dialognachricht noch einmal ausgeben lassen.

Nach Beendigung und erneutem Start der Anwendung kann der Benutzer sich bei der Variante UTM-S nach dem Anmelden mit dem KDCDISP-Kommando die Dialogausgabenachricht vom letzten Sicherungspunkt noch einmal ausgeben lassen.

Gibt der Benutzer das KDCDISP-Kommando nach Abschluss des Anmeldevorgangs oder nach der Rückkehr von einem eingeschobenen Vorgang ein, dann zeigt *open*UTM nochmals den letzten Bildschirm der letzten Session, bzw. den letzten Bildschirm des unterbrochenen Vorgangs.

Bei Bildschirmausgaben in FHS-DE-Formaten lesen Sie bitte auch den Abschnitt „KDCDISP, KDCLAST in FHS-DE-Formaten“ auf Seite 145.

Das KDCDISP-Kommando ist in folgenden Situationen nützlich:

- Aufgrund von Übertragungsproblemen oder durch Fehlbedienung am Terminal ist der Bildschirminhalt nach einer Dialogausgabe teilweise oder vollständig zerstört.
- Der Benutzer hat während der Bearbeitung eines Vorgangs asynchrone Nachrichten am Bildschirm erhalten. Die asynchronen Nachrichten wurden entweder durch *open*UTM direkt gesendet oder der Benutzer hat sie mit KDCOUT selbst angefordert. Wenn der Benutzer danach den offenen Vorgang fortsetzen möchte, lässt er sich dazu mit einem KDCDISP-Kommando die letzte Dialogausgabe nochmals ausgeben.

  Wenn der Benutzer direkt in der asynchronen Formatnachricht den Vorgang fortsetzen will, erzeugt *open*UTM implizit einen KDCDISP, und die Eingabe ist zu wiederholen.
- Nach Beendigung und erneutem Start der UTM-Anwendung kann der Benutzer (zur Orientierung) mit dem KDCDISP-Kommando die letzte Dialogausgabe des vor Beendigung der Anwendung abgeschlossenen Vorgangs wiederholen. Das gilt aber nur, wenn es sich um eine UTM-S-Anwendung handelt und wenn der Vorgangswiederanlauf nicht explizit ausgeschaltet wurde durch die KDCDEF-Anweisung USER ...,RESTART=NO (bzw. LTERM ...,RESTART=NO, wenn die Anwendung ohne Benutzerkennungen generiert wurde).

KDCLAST - Letzte Ausgabe wiederholen

Das Kommando KDCLAST ermöglicht die Wiederholung der letzten Ausgabenachricht am Terminal unabhängig davon, ob es eine Dialognachricht oder eine asynchrone Nachricht war. Bei Bildschirmausgaben in FHS-DE-Formaten sehen Sie auch den folgenden Abschnitt.

Wurde als letztes eine asynchrone Nachricht ausgegeben, so wird diese Ausgabe mit KDCLAST wiederholt. Die asynchrone Nachricht wird dabei aber noch nicht freigegeben.

Wird das Kommando KDCLAST nach Abschluss des Anmeldevorgangs eingegeben, dann zeigt *open*UTM nochmals den letzten Bildschirm des Anmeldevorgangs. Wird es nach der Rückkehr von einem eingeschobenen Vorgang eingegeben, wird der letzte Bildschirm des eingeschobenen Vorgangs ausgegeben.

KDCDISP, KDCLAST in FHS-DE-Formaten

Es wird empfohlen, die Wiederholung der letzten Bildschirmausgabe in FHS-DE-Formaten über die FHS-DE-Funktion RESHOW anzufordern.

Im Zwischendialog muss der Benutzer die Wiederholung der letzten Bildschirmausgabe mit der FHS-DE-Funktion RESHOW anfordern. Die Funktion RESHOW kann nur über eine K-Taste aufgerufen werden, die im zum Format gehörenden KEY-Format mit RESHOW belegt wird. Diese Taste darf in *open*UTM nicht mit SFUNC generiert werden. Siehe dazu auch Benutzerhandbuch „FHS Formatierungssystem für *open*UTM, TIAM, DCAM“.

Wenn die UTM-Kommandos KDCDISP und KDCLAST in FHS-DE-Formaten angegeben werden, dann werden sie von *open*UTM nur bearbeitet, wenn sie in das erste Eingabefeld oder in das UTM-Steuerfeld eines FHS-DE-Bildschirmformats angegeben werden. In dem Bildschirmformat darf kein Feld so belegt sein, dass als Folgeaktion ein FHS-DE-Zwischendialog geführt würde. Sind diese Bedingungen erfüllt, dann wird das Kommando von FHS-DE zur Bearbeitung an *open*UTM übergeben und als Folge die letzte Bildschirmausgabe wiederholt.

Wird eines der Kommandos dagegen in einer Dialog-Box (im FHS-DE-Zwischendialog) angegeben oder in einem Bildschirmformat, dem ein Zwischendialog folgt, dann führen KDCDISP und KDCLAST nicht zur Wiederholung der letzten Bildschirmausgabe durch *open*UTM. Zwischendialoge werden zuerst von FHS-DE bearbeitet. FHS-DE erkennt KDCLAST bzw. KDCDISP in diesem Fall nicht.

KDCOFF - Abmelden von einer UTM-Anwendung

Der Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF von der UTM-Anwendung abmelden. Dadurch wird die Verbindung zum Terminal abgebaut. Er kann danach von diesem Terminal erneut eine Verbindung zu einer UTM-Anwendung aufbauen. Meldet er sich während der Bearbeitung eines Vorgangs am Transaktionsende ab, wird die Bearbeitung unterbrochen. Sie kann nach späterem Anmelden an die UTM-Anwendung wieder fortgesetzt werden.

KDCOFF BUT

Durch Eingabe von KDCOFF BUT kann der Benutzer sich so abmelden, dass die Verbindung zwischen Terminal und UTM-Anwendung bestehen bleibt. Er wird gleich zum erneuten Anmelden aufgefordert, bzw. der Anmeldevorgang wird gestartet.

Meldungen

Nach Eingabe von KDCOFF [BUT] reagiert *open*UTM mit einer der Meldungen:

```
K019 KDCOFF von Anwendung beispiel akzeptiert
```

Der Benutzer hat KDCOFF eingegeben, oder er hat in einer Anwendung ohne Benutzerkennungen KDCOFF BUT eingegeben. Das Terminal ist nicht mehr mit der UTM-Anwendung verbunden.

```
K018 KDCOFF von Anwendung beispiel akzeptiert - Bitte KDCSIGN
```

In einer Anwendung mit Benutzerkennungen und ohne Anmeldevorgang hat der Benutzer KDCOFF BUT eingegeben. *open*UTM fordert ihn zum erneuten Anmelden auf.

```
K003 Das Kommando KDCOFF ist in dieser Situation nicht erlaubt
```

Die Eingabe erfolgte nach einem PEND KP-Aufruf oder blockierenden Aufruf (z.B. PGWT) des Teilprogramms.

9 Programmaustausch im Betrieb

*open*UTM bietet Funktionen, mit denen Sie ein Anwendungsprogramm bzw. Teile eines Anwendungsprogramms im Betrieb austauschen können. Für den Austausch benutzt *open*UTM die Schnittstellen und Funktionen des BLS. Ein Anwendungsprogramm bzw. seine Teile können Sie deshalb nur austauschen, wenn Sie die Anwendung unter Verwendung der BLS-Schnittstelle generiert, gebunden und gestartet haben.

Im laufenden Betrieb können Sie mit Hilfe von UTM-Administrationsfunktionen Folgendes austauschen:

- Alle non-shareable Anwendungsteile, die nicht statisch gebunden sind
- Alle Anwendungsteile in einem Common Memory Pool, der für eine Benutzerkennung gültig ist (MPOOL, ...SCOPE=GROUP). Dies setzt jedoch voraus, dass sich an den Common Memory Pool nicht mehrere UTM-Anwendungen angeschlossen haben.
- Die komplette Anwendung.
- Ein nicht-privilegiertes Subsystem im LLM-Format, bei dem der Private Slice als Lademodul mit LOAD-MODE=STARTUP | ONCALL generiert ist. Es ist zusätzlich zum KDCPROG-Kommando noch ein Systemadministrationskommando nötig. Es ist zu beachten, dass für nicht-privilegierte Subsysteme im OM-Format eine Konsistenzlücke existiert. Sehen Sie dazu die Hinweise im Abschnitt „Shared Code im Systemspeicher“ auf Seite 26.

Nicht im laufenden Betrieb austauschbar sind:

- Programmteile, die statisch zum Anwendungsprogramm gebunden wurden.
- Programmteile, die als shareable Programme in einen Common Memory Pool geladen wurden, der mit SCOPE=GLOBAL generiert ist.
- Programmteile in Common Memory Pools, die mit SCOPE=GROUP geladen wurden und an die sich mehrere UTM-Anwendungen unter derselben Benutzerkennung anschließen.
- Lademodule, in denen neben dem Lademodul zusätzliche benutzereigene Module per Autolink-Funktion geladen werden.
- Lademodule, die TCBs enthalten.

- Lademodule, die Laufzeitmodule enthalten. Diese können nur mit dem gesamten Anwendungsprogramm ausgetauscht werden.
- Das Lademodul, das das Administrationsteilprogramm KDCADM enthält.



Die UTM-Administrationsfunktionen zum Programmaustausch sind im *open*UTM-Handbuch „Anwendungen administrieren“ beschrieben.

9.1 Binden und Generieren

Damit Sie Anwendungsteile austauschen können, muss das Start-LLM als Bindelademodul (LLM) vorgebunden und in einer Programmbibliothek als Element vom Typ L bereitgestellt werden (siehe Abschnitt „Binden von LLMs“ auf Seite 41).

Die übrigen Programmteile müssen entweder ebenfalls als Bindelademodul (LLM) vorgebunden und in einer Programmbibliothek als Elemente vom Typ L bereitgestellt werden, oder sie müssen als Bindemodul (OM) in einer Bindemodulbibliothek (OML) oder in einer Programmbibliothek als Elemente vom Typ R stehen.

Austauschbare Lademodule müssen immer in einer LOAD-MODULE-Anweisung mit LOAD-MODE≠STATIC generiert sein. Falls sie in einen Common Memory Pool geladen werden, dann muss dieser mit MPOOL ,...SCOPE=GROUP generiert sein. Werden Teilprogramme oder Datenbereiche in ein Lademodul eingebracht, dann muss in den zugehörigen PROGRAM- oder AREA-Anweisungen der Operand LOAD-MODULE entsprechend versorgt werden.

9.2 Anwendungsteile austauschen

Einen Austausch von Anwendungsteilen müssen Sie explizit über das Administrationskommando KDCPROG anfordern. Dabei müssen Sie die Versionsnummer des neu zu ladenden Lademoduls angeben. *open*UTM prüft die Zulässigkeit der Angaben und veranlasst den Programmaustausch. *open*UTM verifiziert beim Programmaustausch nicht, ob die mit KDCDEF definierte Zuordnung in der LOAD-MODULE-, AREA- und PROGRAM-Anweisung der tatsächlichen Aufteilung der Lademodule in den Bibliotheken entspricht.

Die Änderungen, die sich durch die Administrationsaktionen zum Programmaustausch in dem geladenen Anwendungsprogramm ergeben haben, werden von *open*UTM über das Ende des Anwendungslaufs hinaus gesichert; d.h. beim nächsten Start werden die durch Programmaustausch geladenen Versionen der Lademodule gestartet. Auch bei einer

Änderungsgenerierung mittels KDCUPD können die Versionsnummern der ausgetauschten Lademodule in die neue KDCFILE übernommen werden, so dass bei dem nächsten Start der Anwendung wieder die zuletzt geladenen Module geladen werden.

Als Einzelmodul kann beim Programmaustausch nur ein Programmteil ausgetauscht werden, der mit **einem** Ladevorgang geladen wurde; d.h. als Teil kann jeweils **nur ein** LLM oder OM ausgetauscht werden. Es wird nur das Modul ausgetauscht, nicht die gesamte Ladeeinheit, d.h. wenn die Ladeeinheit Teile des Laufzeitsystems enthält, die per Autolink dazugeladen wurden, so werden diese beim Austausch nicht entladen.

Beispiel

Das Lademodul A-LLM steht in der Bibliothek OWN-LIB und ist generiert mit

```
LOAD-MODULE A-LLM,LIB=OWN-LIB                                  -
                 ,VERSION=001                                  -
                 ,LOAD-MODE=STARTUP                            -
                 .ALTERNATE-LIBRARIES=YES
```

A-LLM enthält beispielsweise ein Teilprogramm APU, das eine Funktion *bfunc* aufruft. Diese Funktion ist in B-LLM, das sich weder in A-LLM noch in der Bibliothek OWN-LIB befindet. In diesem Fall wird A-LLM beim Start des Anwendungsprogramms mit B-LLM per Autolink-Funktion geladen. Bei einem Austausch des A-LLM mit KDCPROG bleibt B-LLM mit *bfunc* im Speicher. Dies kann zu Inkonsistenzen führen, wenn B-LLM Unterprogramme von Anwendungsprogrammen und nicht ausschließlich Laufzeitmodule enthält. Enthält B-LLM auch Anwendungslogik, dann sollten A-LLM und B-LLM zu einem LLM gebunden werden.

Bei einem Austausch von einzelnen Lademodulen werden die Event-Exits SHUT bzw. START nicht durchlaufen. Diese werden nur dann aktiviert, wenn bedingt durch einen Programmaustausch in einer Task das gesamte Anwendungsprogramm beendet und neu geladen wird.

Der Austausch verläuft unterschiedlich, je nachdem, zu welchem Zeitpunkt (STARTUP oder ONCALL) und wohin (Common Memory Pool oder nicht) das Lademodul geladen wird.

9.2.1 Lademodul mit LOAD-MODE=STARTUP austauschen

Beim Austausch eines Lademoduls, das mit LOAD-MODE=STARTUP generiert wurde, wird der Programmlauf in den Tasks der Anwendung nicht beendet. Vielmehr wird das betroffene Lademodul entladen und anschließend eine neue Version dieses Lademoduls geladen. Diesen Programmaustausch können mehrere Tasks einer Anwendung gleichzeitig ausführen. Während des Austauschs sind in den Tasks der UTM-Anwendung unterschiedliche Stände des Anwendungsprogramms geladen. Jede Task der Anwendung führt

den angeforderten Programmaustausch am Ende der Bearbeitung des aktuellen Auftrags durch. Der Programmaustausch ist abgeschlossen, wenn in jeder Task der Anwendung die neue Version des Lademoduls geladen ist.

Bis der Programmaustausch in allen Tasks abgeschlossen ist, darf kein weiterer Programmaustausch gestartet werden. Der Administrator kann sich mit dem Administrationskommando KDCINF SYSP darüber informieren, ob zur Zeit ein Programmaustausch läuft.

Die Versionsnummern von altem und neuem Lademodul dürfen übereinstimmen.

Bei erfolgreichem Abschluss des Programmaustauschs erzeugt *open*UTM eine K074-Meldung, die nach SYSOUT ausgegeben wird. Die Meldung kann aber auch über ein MSGTAC-Programm auswertet werden, um dem Administrator diese Information zugänglich zu machen.

Muss der Programmaustausch durch *open*UTM abgebrochen werden, dann erzeugt *open*UTM eine K075-Meldung.

9.2.2 Lademodul mit LOAD-MODE=ONCALL austauschen

Soll ein Teil einer Anwendung ausgetauscht werden, der in einem Lademodul mit LOAD-MODE=ONCALL generiert wurde, dann wird bei der Bearbeitung des Administrationskommandos KDCPROG nur die neu zu ladende Versionsnummer des betroffenen Lademoduls in die UTM-Tabellen eingetragen.

Das Laden des Lademoduls der neuen Version wird von jeder Task der Anwendung erst dann veranlasst, wenn in der Task das nächste Mal ein Teilprogramm dieses Lademoduls aufgerufen wird. Diesen Programmaustausch können mehrere Tasks einer Anwendung gleichzeitig ausführen. Bis der angeforderte Programmaustausch von allen Tasks der UTM-Anwendung durchgeführt wurde, sind in den einzelnen Tasks unterschiedliche Stände des Anwendungsprogramms geladen. Es ist jedoch sichergestellt, dass jede Task den angeforderten Austausch durchführt, bevor erneut ein Teilprogramm aktiviert wird, das in dem auszutauschenden Lademodul enthalten ist.

Der Austausch eines ONCALL-Lademoduls wirkt nicht blockierend auf nachfolgende Kommandos zum Programmaustausch; d.h. der Administrator kann unmittelbar nach der Bearbeitung des Kommandos KDCPROG mit einem weiteren Kommando KDCPROG einen neuen Programmaustausch veranlassen. Sie dürfen jedoch die Programmbibliothek nach dem Administrationsaufruf nicht mehr verändern, da andernfalls der Programmaustausch zu Fehlern führen kann.

Stimmen die Versionsbezeichner von neuem und altem Lademodul überein, wird kein Programmaustausch durchgeführt.

9.2.3 Lademodul in einem Common Memory Pool austauschen

Lademodule, die ganz oder teilweise in Common Memory Pools liegen, können mit dem Kommando KDCPROG zunächst zum Austausch vorgemerkt werden. Ausgetauscht werden diese Lademodule erst bei einem nachfolgenden Austausch der gesamten Anwendung, siehe unten.

Wird die Anwendung vor dem Anwendungstausch beendet, dann *bleiben* die Lademodule zum Austausch vorgemerkt. Beim nachfolgenden Neustart werden wieder die alten Versionen geladen.

Bei Anforderung eines Programmaustauschs über KDCPROG muss die Versionsnummer des neu zu ladenden Lademoduls angegeben werden. Die Versionsnummern von altem und neuem Lademodul dürfen übereinstimmen.

Teilprogramme, die sowohl shareable- als auch non-shareable-Teile enthalten, sollten als LLM gebunden werden, da sonst zwei Lademodule im OM-Format existieren, die nicht gleichzeitig ausgetauscht werden können (Konsistenzproblem!).

9.3 Gesamte Anwendung austauschen

Mit dem Kommando KDCAPPL PROG=NEW können Sie die gesamte Anwendung austauschen. Dies ist in den folgenden Fällen sinnvoll bzw. notwendig:

- Programmteile in Common Memory Pools sollen ausgetauscht werden.
- Programmteile, die mit LOAD-MODE=ONCALL generiert sind, sollen entladen werden.
- Programme in Common Memory Pools wurden dynamisch hinzugenommen, und diese Programme sollen für die Anwendung geladen werden.

Beim Austausch der gesamten Anwendung wird nacheinander jede Task der Anwendung entladen und anschließend neu geladen. Beim Neuladen werden die neuen Versionen der Lademodule geladen. Um den Betrieb der Anwendung möglichst wenig zu stören, tauscht zu einer Zeit immer nur eine Task der Anwendung aus.

Der Administrator kann sich mit dem Kommando KDCINF SYSP darüber informieren, ob zur Zeit ein Programmaustausch läuft.

Bei erfolgreichem Abschluss des Programmaustauschs erzeugt *open*UTM eine K074-Meldung, die nach SYSOUT ausgegeben wird. Die Meldung kann aber auch über ein MSGTAC-Programm auswertet werden, um dem Administrator diese Information zugänglich zu machen.

Können einzelne Lademodule nicht geladen werden oder sind nicht alle Programme, entgegen den Angaben bei der Generierung, in den Lademodul gebunden, dann führt dies nicht zum Abbruch des Anwendungsaustausches. Beim späteren Aufruf eines nicht verfügbaren Programms erzeugt *open*UTM einen PEND ER.

Muss der Programmaustausch durch *open*UTM abgebrochen werden, dann erzeugt *open*UTM eine K075-Meldung. Die K075-Meldung enthält als Insert die TSN der Task, die den Anwendungsaustausch abgebrochen hat. In der SYSOUT-Datei dieser Task kann nach Meldungen des BLS und Meldungen von *open*UTM gesucht werden, die Hinweise auf den Grund des Abbruchs geben. Ein Grund für den Abbruch eines Anwendungsaustausches kann beispielsweise sein, dass eine AREA in einem neugeladenen Lademodul nicht verfügbar ist. Wird der Anwendungsaustausch abgebrochen, dann wird evtl. eine Task der UTM-Anwendung beendet. Die anderen Tasks der Anwendung laufen jedoch weiter. Die zum Austausch vorgemerkten Lademodule bleiben vorgemerkt, und sobald das Problem behoben ist, kann der Anwendungsaustausch erneut angestoßen werden.

9.4 Programme dynamisch hinzufügen

Per dynamischer Administration können u.a. Programme im Betrieb der Anwendung neu generiert werden. Näheres zur dynamischen Administration siehe *open*UTM-Handbuch „Anwendungen administrieren“.

Bevor diese Programme aufgerufen werden können, müssen sie zunächst geladen werden. Dazu muss das Programm zu dem zugeordneten Lademodul gebunden und mit einer neuen Version in der Programmbibliothek bereitgestellt werden, die in der LOAD-MODULE-Anweisung beim Generieren angegeben wurde.

Anschließend muss der Administrator dieses Lademodul durch das Kommando KDCPROG oder durch Programmaufruf austauschen. Falls das Lademodul in einem Common Memory Pool liegt, muss die gesamte Anwendung ausgetauscht werden, d.h. zunächst wird das Modul zum Austausch vorgemerkt (z.B. mit KDCPROG) und anschließend wird die gesamte Anwendung ausgetauscht (z.B. mit KDCAPPL PROG=NEW).

10 Fehlertoleranz von *open*UTM

Fehlertoleranz heißt hier, dass eine UTM-Anwendung auch dann betriebsbereit bleibt, wenn in einzelnen Teilprogrammen Fehler auftreten, die *open*UTM zum Abbruch einer Transaktion zwingen. *open*UTM sorgt dann dafür, dass das Anwendungsprogramm beendet und neu geladen wird, so dass sich der Fehler nicht weiter ausbreitet und andere Benutzer der Anwendung und deren Daten dadurch nicht beeinträchtigt werden.

Beim Fehlerverhalten von *open*UTM unterscheidet man:

- Interne UTM-Fehler und Fehler in der Systemumgebung

  Sie führen zum abnormalen Beenden der Anwendung, ebenso wie das Administrationskommando KDCSHUT KILL oder das Absetzen eines KDCADMI-Aufrufs mit Operationscode KC_SHUTDOWN und Subcode KC_KILL.
  *open*UTM erzeugt für jeden Prozess der Anwendung einen UTM-Dump. Der UTM-Dump wird mit dem Tool KDCDUMP aufbereitet. Wie das geht, ist im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“ beschrieben.

- Fehler im Anwendungsprogramm

  Dabei handelt es sich um Fehler in Teilprogrammen, die sich in zwei Gruppen einteilen lassen:

  – Fehler, die zum Neuladen des Anwendungsprogramms führen

  – Fehler, die gegebenenfalls eine Fortsetzung des Programms erlauben

10.1 Fehler, die *open*UTM erkennt

Ein Teilprogramm wird in folgenden Fällen durch *open*UTM abnormal beendet:

- Es wurde ein PEND ER oder FR programmiert.
- Ein UTM-Aufruf hat einen KDCS-Returncode ≥ 70Z geliefert. In diesem Fall setzt *open*UTM intern PEND ER.

In allen Fällen bricht *open*UTM also den Vorgang ab. Falls PEND FR programmiert war, veranlasst *open*UTM keine weitere Aktionen.

Falls der Vorgang durch PEND ER (per Programm oder intern) beendet wurde, erzeugt *open*UTM

- einen UTM-Dump mit REASON=PENDER, der nur die Daten des KDCROOT wiedergibt und
- einen Speicherauszug des Klasse 5- und Klasse 6-Speichers. Dieser ist mit dem BS2000-Dienstprogramm DAMP aufzubereiten.

  Standardmäßig wird dieser Speicherauszug unterdrückt, da *open*UTM als ENTER-Prozess läuft. Möchten Sie Speicherauszüge des Klasse-5- und Klasse-6-Speichers erzeugt haben, geben Sie in der Startprozedur (siehe Seite 92) vor dem Laden des Anwendungsprogramms das Kommando

  ```
  /MODIFY-TEST-OPTIONS DUMP=YES
  ```

Anschließend beendet *open*UTM das betroffene Anwendungsprogramm. Dadurch wird verhindert, dass Folgefehler wegen eines eventuell überschriebenen Anwendungsprogramms entstehen.

Mit einer Sprunganweisung in der Startprozedur lässt sich erreichen, dass das Anwendungsprogramm neu geladen wird und *open*UTM mit der gewünschten Anzahl Tasks weiterläuft. Fehlt die Sprunganweisung, wird die Task (siehe Seite 92) und bei der letzten Task die Anwendung beendet.

10.2 Fehler, die das BS2000 erkennt und zu einem STXIT-Ereignis führen

In *open*UTM sind für die folgenden Ereignisklassen STXIT-Routinen definiert:

– PROCHK
– TIMER
– ERROR
– ABEND
– TERM

Bei Eintritt der STXIT-Ereignisse PROCHK, TIMER und ERROR ermöglicht *open*UTM dem Sprachanschluss-Modul des zuletzt aktiven Teilprogramms alternativ:

– Diagnoseinformationen ausgeben und das Teilprogramm abnormal beenden, oder
– das Teilprogramm definiert fortsetzen (z.B. über ON-Condition in PL/I)

Tritt ein STXIT-Ereignis ABEND oder TERM ein, setzt *open*UTM PEND ER, beendet das Anwendungsprogramm mit TERM und schreibt Folgendes nach SYSOUT:

– die Registerinhalte zum Zeitpunkt des Ereignisses
– die Befehlszähler zum Unterbrechungszeitpunkt
– das Unterbrechungsgewicht.

Waren die zuletzt aktiven Teilprogramme COBOL-, Assembler- oder SPL-Teilprogramme, so werden alle STXIT-Ereignisse behandelt wie TERM und ABEND.

Zu STXIT-Ereignissen siehe Handbuch „BS2000/OSD Makroaufrufe an den Ablaufteil“. Eine Tabelle zur Zuordnung von Ereignis zu Unterbrechungsgewicht finden Sie im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.

Benutzerdefinierte STXIT-Routinen

In den Teilprogrammen können den Ereignissen PROCHK und ERROR eigene STXIT-Ausgänge zugeordnet werden. Dabei müssen mit dem Operanden STXDNEW eigene Verwaltungsblöcke für diese Ereignisse angelegt werden.

Diese benutzerdefinierten STXIT-Routinen müssen mit dem Makroaufruf EXIT CONTINU=YES beendet werden, damit *open*UTM seine eigene (und gleichzeitig letzte) STXIT-Routine derselben Ereignisklasse starten kann.

11 SAT-Protokollierung

Sicherheitsrelevante UTM-Ereignisse können mit der BS2000-Funktion SAT (**S**ecurity **A**udit **T**rail) protokolliert werden. SAT dient u.a. zur Beweissicherung bei unerlaubten Eindringversuchen und ermöglicht es dadurch, sofort auf derartige Ereignisse zu reagieren (Alarmfunktion). Dadurch kann ein eventueller Schaden gering gehalten oder ganz vermieden werden.

Voraussetzung dafür sind die BS2000-Komponente SECOS und das Subsystem SATCP. *open*UTM bietet Ihnen die Möglichkeit, die SAT-Protokollierung von UTM-Ereignissen für Ihre Anwendung zu steuern: durch die KDCDEF-Generierung (MAX-Anweisung) und durch die UTM-SAT-Administration (siehe ab Seite 171).

Für eine UTM-Anwendung kann die SAT-Protokollierung durch die Generierung (MAX ...,SAT=ON) oder durch die UTM-SAT-Administration (KDCMSAT=ON) eingeschaltet werden. Per UTM-SAT-Administration kann die Protokollierung im laufenden Betrieb jederzeit wieder ausgeschaltet werden (KDCMSAT SAT=OFF). Bei eingeschalteter SAT-Protokollierung wird eine Mindestprotokollierung durchgeführt.
Die Mindestprotokollierung umfasst die folgenden UTM-Ereignisse:

– An- und Abmelden einer Task bei der UTM-Anwendung
– Eingabe eines UTM-SAT-Administrationskommandos
– Austausch von Programmteilen mit Hilfe des BLS

Zusätzlich können weitere Ereignisse definiert werden. Die Protokollierung dieser Ereignisse kann ereignisspezifisch, benutzerspezifisch und auftragsspezifisch ein- und ausgeschaltet werden. Das Vordefinieren der zu protokollierenden Ereignisse nennt man Preselection (siehe Seite 159). Die Preselection kann per Generierung und per UTM-SAT-Administration erfolgen.

Der Aufbau der SAT-Protokollsätze ist im Anhang ab Seite 255 beschrieben.



ACHTUNG!
Der BS2000-Sicherheitsbeauftragte (BS2000-Benutzerkennung $SYSPRIV bei Auslieferung) kann die SAT-Protokollierung unterdrücken. Siehe BS2000/OSD-Handbuch „SECOS“. Der SAT-Administrator der UTM-Anwendung muss sich daher mit dem BS2000-Sicherheitsbeauftragten absprechen. Bei nicht erfolgter Absprache wird die Anwendung abnormal beendet und die Meldung K126 ausgegeben, siehe *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“.

11.1 Sicherheitsrelevante UTM-Ereignisse

Es sind elf sicherheitsrelevante UTM-Ereignisse definiert, die protokolliert werden können. Für jedes Ereignis wird ein Ergebnis (Result) protokolliert: Erfolg (Success) oder Misserfolg (Failure). Erfolgreich heißt z.B. beim Anmelden eines Benutzers, dass *open*UTM die angegebene Benutzerkennung und die Authentisierungsdaten akzeptiert hat.

Der folgenden Tabelle können Sie entnehmen, welche UTM-Ereignisse es gibt und bei welchem Ergebnis sie protokolliert werden können.

Name des Ereignisses	**Bedeutung**	**Protokollierung möglich**
TASK-ON	Anschließen einer Task an die UTM-Anwendung	Erfolg
TASK-OFF	Abmelden einer Task von der UTM-Anwendung	Erfolg
SIGN	Anmelden eines UTM-Benutzers	Erfolg / Misserfolg
CHANGE-PW	Ändern des Benutzerkennworts	Erfolg / Misserfolg
START-PU	Erzeugen eines Auftrags bzw. Start eines Teilprogrammlaufes	Erfolg / Misserfolg
END-PU	Beendigung eines Teilprogrammlaufes	Erfolg
DATA-ACCESS	Zugriff auf einen vorgangsübergreifenden UTM-Speicherbereich (ULS, GSSB, TLS)	Erfolg / Misserfolg
ADM-CMD	Ausführung eines Administrationsaufrufs	Erfolg / Misserfolg
SEL-CMD	Ausführung eines UTM-SAT-Administrationskommandos	Erfolg / Misserfolg
CHG-PROG	Austausch von Lademodulen durch BLS	Erfolg / Misserfolg

Hinweise

- Das Ereignis SEL-CMD wird als einziges auch dann protokolliert, wenn die SAT-Protokollierung für die UTM-Anwendung ausgeschaltet ist, sofern die Generierung des BS2000/OSD eine SAT-Protokollierung zulässt.
- Die Ereignisse CHG-ACC-KEY, TASK-ON, TASK-OFF und CHG-PROG werden immer protokolliert, wenn die Protokollierung für die UTM-Anwendung eingeschaltet ist (Mindestprotokollierung).
- Das Ereignis "Transaktionsende" (END-PU mit TACNAID=T oder C) wird immer protokolliert, wenn die Protokollierung für END-PU nicht explizit durch ereignisspezifische Preselection ausgeschaltet (OFF in der Anweisung SATSEL) und wenn für diese Transaktion mindestens ein anderes Ereignis protokolliert wurde.
- Die anderen Ereignisse aus der Tabelle werden erst protokolliert, wenn sie durch Preselection definiert wurden und die Protokollierung eingeschaltet ist.
- Alle UTM-Ereignisse können mit der ALARM-Funktion von SAT gekoppelt werden. Beim Auftreten des Ereignisses wird dann eine Meldung an der Konsole des BS2000/OSD-Rechners ausgegeben.

11.2 Preselection - zu protokollierende Ereignisse einstellen

Die Ereignisse SIGN, CHANGE-PW, START-PU, END-PU mit TACAID=P, DATA-ACCESS und ADM-CMD werden auch bei eingeschalteter Protokollierung nicht automatisch protokolliert. Die Protokollierung muss zusätzlich ereignisspezifisch eingeschaltet werden. Die Steuerung der Protokollierung dieser Ereignisse wird Preselection genannt. Die Protokollierung kann ereignisspezifisch, benutzerspezifisch und auftragsspezifisch gesteuert werden. Die Preselection kann per Generierung und per UTM-SAT-Administration erfolgen.

Wenn die Preselection-Werte per Administration eingestellt werden, dann gilt:

- Die Preselection-Werte für ereignisgesteuerte Protokollierung gelten nur für die Dauer des Anwendungslaufs. Bei jedem Anwendungsstart werden wieder die generierten Werte eingestellt.
- Die Preselection-Werte für benutzer- und auftragsspezifische Protokollierung bleiben dauerhaft wirksam, auch bei UTM-F. Sie können mit dem inversen KDCDEF in eine neue Generierung übernommen werden.

11.2.1 Ereignisgesteuerte SAT-Protokollierung

Für die Ereignisse SIGN, CHANGE-PW, START-PU, END-PU, DATA-ACCESS und ADM-CMD kann die SAT-Protokollierung jeweils einzeln ein- und ausgeschaltet werden. Für das Ereignis DATA-ACCESS (Zugriff auf UTM-Speicherbereich) kann die Protokollierung für die Speichertypen GSSB, TLS und ULS jeweils einzeln gesteuert werden.

Sie können für jedes einzelne Ereignis angeben:

OFF	Das Ereignis wird nie protokolliert, auch wenn die benutzer- bzw. auftragsspezifische Protokollierung eingeschaltet ist.
SUCC	Das Ereignis wird im Erfolgsfall protokolliert.
FAIL	Das Ereignis wird protokolliert, falls es nicht erfolgreich war.
BOTH	Das Ereignis wird unabhängig vom Ergebnis protokolliert.
NONE	Keine ereignisgesteuerte Protokollierung

Die Preselection-Werte für die ereignisgesteuerte Protokollierung können Sie bei der Generierung einstellen durch die Anweisung SATSEL. Per UTM-SAT-Administration können Sie die Preselection-Werte einstellen mit dem Kommando:

```
KDCMSAT SATSEL=...,EVENT=(...)
```

Diese Einstellung gilt dann nur für die Dauer des Anwendungslaufs.

Beispiel

Das Ereignis "Kennwort ändern" (CHANGE-PW) soll protokolliert werden, im Erfolgsfall (Änderung wurde von *open*UTM akzeptiert) und bei Misserfolg. Das Absetzen eines Administrationskommandos soll bei einem Misserfolg protokolliert werden.

Generierung:

```
SATSEL BOTH,EVENT=CHANGE-PW
SATSEL FAIL,EVENT=ADM-CMD
```

Administration:

```
KDCMSAT SATSEL=BOTH,EVENT=CHANGE-PW
KDCMSAT SATSEL=FAIL,EVENT=ADM-CMD
```

11.2.2 Benutzergesteuerte SAT-Protokollierung

Für jeden einzelnen UTM-Benutzer kann definiert werden, ob die von ihm ausgelösten Ereignisse SIGN, CHANGE-PW, START-PU, END-PU, DATA-ACCESS, ADM-CMD und eventuell sicherheitsrelevante Ereignisse einer beteiligten Datenbank protokolliert werden sollen. Die Ereignisse werden für den Benutzer jedoch nicht protokolliert, wenn die SAT-Protokollierung für dieses Ereignis mit OFF ausgeschaltet ist.
Generierung: `SATSEL OFF,EVENT=...`
UTM-SAT-Administration: `KDCMSAT SATSEL=OFF,EVENT=...`

Sie können für jeden Benutzer angeben:

SUCC	Die Ereignisse, die der Benutzer auslöst, werden im Erfolgsfall protokolliert.
FAIL	Die Ereignisse, die der Benutzer auslöst, werden protokolliert, falls sie nicht erfolgreich waren.
BOTH	Die Ereignisse, die der Benutzer auslöst, werden unabhängig vom Ergebnis protokolliert.
NONE	Keine benutzergesteuerte Protokollierung

Die Preselection-Werte für die benutzergesteuerte Protokollierung können Sie bei der Generierung einstellen durch die Anweisung

`USER` *username*`,...,SATSEL=...`

Per UTM-SAT-Administration können Sie die Preselection-Werte einstellen mit dem Kommando:

`KDCMSAT SATSEL=...,USER=`*username*

Für SATSEL kann einer der Werte BOTH, SUCC, FAIL oder NONE angegeben werden.

Diese Einstellung bleibt über das Anwendungsende hinaus erhalten, auch bei UTM-F.

11.2.3 Auftragsgesteuerte SAT-Protokollierung

Für jeden Transaktionscode kann definiert werden, ob die von dem zugehörigen Teilprogramm ausgelösten Ereignisse CHANGE-PW, START-PU, END-PU, DATA-ACCESS, ADM-CMD und evtl. sicherheitsrelevante Ereignisse einer beteiligten Datenbank protokolliert werden sollen. Zusätzlich wird auch das Erzeugen von Aufträgen dieses Transaktionscodes (START-PU) protokolliert. Ein Ereignis wird für den Transaktionscode jedoch nicht protokolliert, wenn die SAT-Protokollierung für dieses Ereignis mit OFF ausgeschaltet ist.

Sie können für jeden Transaktionscode angeben:

SUCC Die Ereignisse des Teilprogrammlaufes werden im Erfolgsfall protokolliert.

FAIL Die Ereignisse des Teilprogrammlaufes werden protokolliert, falls sie nicht erfolgreich waren.

BOTH Die Ereignisse des Teilprogrammlaufes werden unabhängig vom Resultat protokolliert.

NONE Keine auftragsgesteuerte Protokollierung

Die Preselection-Werte für die auftragsgesteuerte Protokollierung können Sie bei der Generierung einstellen durch die Anweisung:

```
TAC tacname,...,SATSEL=...
```

Per UTM-SAT-Administration können Sie die Preselection-Werte einstellen mit dem Kommando:

```
KDCMSAT SATSEL=...,TAC=tacname
```

Diese Einstellung bleibt über das Anwendungsende hinaus erhalten, auch bei UTM-F.

11.2.4 Preselection-Werte voreinstellen

Bei der Generierung können Sie Preselection-Werte angeben, ohne die SAT-Protokollierung einzuschalten. Die Angaben dienen in diesem Fall als Voreinstellung für die SAT-Protokollierung, die per UTM-SAT-Administration im laufenden Betrieb bei Bedarf eingeschaltet werden kann. Die definierten Werte können durch die UTM-SAT-Administration dann noch verändert werden. Änderungen für die ereignisgesteuerte Protokollierung gelten jedoch nur für die Dauer des aktuellen Anwendungslaufs.

Voreinstellung bei der Generierung:

```
MAX ...,SAT=OFF
SATSEL ...
TAC ...,SATSEL=...
USER ...,SATSEL=...
```

Einschalten per UTM-SAT-Administration:

```
KDCMSAT SAT=ON
```

11.2.5 Preselection-Werte verknüpfen

Werden für Ereignisse mehrere Preselection-Werte eingestellt (ereignis-, benutzer-, auftragsgesteuert), so ist das Ergebnis die Oder-Verknüpfung aus den einzelnen Preselection-Werten.

Die folgenden Tabellen zeigen die möglichen Kombinationen von SAT-Protokollierungsbedingungen.

Bedeutung der Spalten:

EVENT ereignisspezifisch eingestellte Protokollierung
(Generierung SATSEL ...;
UTM-SAT-Administration KDCMSAT SATSEL=...,EVENT=...)

TAC auftragsspezifisch eingestellte Protokollierung
(Generierung TAC ...,SATSEL=...;
UTM-SAT-Administration KDCMSAT SATSEL=...,TAC=...)

USER benutzerspezifisch eingestellte Protokollierung
(Generierung USER ...,SATSEL=...;
UTM-SAT-Administration KDCMSAT SATSEL=...,USER=...)

Result Ergebnis der Oder-Verknüpfung, d.h. der Kombination der Preselection-Werte.

Ist für ein Ereignis EVENT der Wert OFF (genereller Verzicht auf die Protokollierung des Ereignisses) generiert, so wird der Wert OFF als Ergebnis übergeben.

EVENT	SUCC															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE
Result	SUCC	BOTH	BOTH	SUCC	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	SUCC	BOTH	BOTH	SUCC

EVENT	FAIL															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE
Result	BOTH	BOTH	BOTH	BOTH	BOTH	FAIL	BOTH	FAIL	BOTH	BOTH	BOTH	BOTH	BOTH	FAIL	BOTH	FAIL

EVENT	BOTH															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE
Result	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH	BOTH

EVENT	NONE															
USER	SUCC				FAIL				BOTH				NONE			
TAC	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE	SUCC	FAIL	BOTH	NONE
Result	SUCC	BOTH	BOTH	SUCC	BOTH	FAIL	BOTH	FAIL	BOTH	BOTH	BOTH	BOTH	SUCC	FAIL	BOTH	NONE

Beispiel für die Verknüpfung der Preselection-Werte

Es werden folgende Preselection-Werte generiert:

Ereignisspezifisch:

```
SATSEL FAIL,EVENT=(SIGN,TLS)
SATSEL SUCC,EVENT=(CHANGE-PW,GSSB,ADM-CMD)
SATSEL BOTH,EVENT=(START-PU,ULS)
SATSEL NONE,EVENT=END-PU
```

Benutzerspezifisch:

```
USER BSPUSER,SATSEL=FAIL
```

Auftragsspezifisch:

```
TAC BSPTAC,SATSEL=SUCC
```

Daraus ergeben sich als Ergebnis die folgenden Preselection-Werte. Die Preselection-Werte werden mit dem UTM-SAT-Administrationskommando KDCISAT abgefragt.

Ereignisspezifisch mit dem Kommando KDCISAT:

SAT	USER	TAC	EVENT	RESULT
OFF			SIGN	FAIL
			CHANGE-PW	SUCC
			START-PU	BOTH
			END-PU	NONE
			GSSB	SUCC
			TLS	FAIL
			ULS	BOTH
			ADM-CMD	SUCC

Benutzerspezifisch mit dem Kommando KDCISAT USER=BSPUSER:

SAT	USER	TAC	EVENT	RESULT
OFF	BSPUSER		SIGN	FAIL
			CHANGE-PW	BOTH
			START-PU	BOTH
			END-PU	FAIL
			GSSB	BOTH
			TLS	FAIL
			ULS	BOTH
			ADM-CMD	BOTH

Auftragsspezifisch mit dem Kommando KDCISAT TAC=BSPTAC:

SAT	USER	TAC	EVENT	RESULT
OFF		BSPTAC	SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH SUCC BOTH SUCC SUCC BOTH BOTH SUCC

11.3 Regeln der SAT-Protokollierung

- Die Ereignisse CHG-ACC-KEY, TASK-ON, TASK-OFF, CHG-PROG und mit Einschränkung END-PU (siehe Abschnitt „Sicherheitsrelevante UTM-Ereignisse“ auf Seite 158) werden genau dann protokolliert, wenn die SAT-Protokollierung für die UTM-Anwendung nicht ausgeschaltet ist.
- Alle anderen Ereignisse werden genau dann protokolliert, wenn die Protokollierung für das jeweilige Ereignis nicht ausgeschaltet ist und zusätzlich wenigstens eines der drei Kriterien ereignisspezifische, benutzerspezifische oder auftragsspezifische Preselection für dieses Ereignis erfüllt ist.
- Ein Ereignis wird von *open*UTM als erfolgreich protokolliert (SUCC), wenn es die UTM-Rechteprüfung erfolgreich durchlaufen hat und die Aktion von *open*UTM auch durchgeführt werden kann. Andernfalls wird das Ereignis als nicht erfolgreich (FAIL) protokolliert.
- UTM-Ereignisse, die durch KDCS-Aufrufe ausgelöst werden (z.B. DATA-ACCESS), werden nicht protokolliert, falls die Funktion wegen ungültigem Wert von KCOM, KCLA, KCLM usw. nicht ausgeführt werden kann (siehe *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“). Die fehlerhafte Versorgung dieser Bereiche kann nur durch ein fehlerhaftes Teilprogramm erfolgen.
- UTM-Ereignisse, die durch KDCS-Aufrufe ausgelöst werden, werden als Misserfolg protokolliert, wenn die Versorgung von KCRN, KCUS oder KCLT fehlerhaft ist. Die fehlerhafte Versorgung von KCRN, KCUS und KCLT kann nicht nur durch ein fehlerhaftes UTM-Teilprogramm, sondern auch durch einen UTM-Benutzer verursacht werden (z.B. wenn dem Benutzer die Berechtigung für die aufgerufene Funktion fehlt).
- Anhand der SAT-Protokollsätze können Sie erkennen, ob ein Ereignis durch Rücksetzen der Transaktion nachträglich unwirksam wurde:
  - Der Protokolldatensatz des betreffenden Ereignisses enthält die Identfikationsnummer der Transaktion, die das Ereignis ausgelöst hat (Protokollfeld UTMTAID).
  - Über die Transaktionsidentifikation finden Sie den zugehörenden Protokollsatz "Transaktionsende" (END-PU im Protokollfeld UTMSUBC). Dort ist im Feld UTMSTAT der Transaktionsstatus angegeben.

  Der Aufbau der Protokollsätze ist im Abschnitt „Aufbau der SAT- Protokollsätze“ im Anhang (Seite 255) beschrieben.
- Der KDCS-Aufruf RSET (Rücksetzen der Transaktion bei Fortsetzung des Teilprogrammlaufes) löst implizit zuerst das Ereignis "Transaktionsende" mit Transaktionsstatus "Rollback" (UTMOBJ2= oder UTMSTAT= R) und dann das Ereignis "Beginn einer Transaktion" (START-PU) aus.

11.4 Postselection - Protokollsätze auswerten

Die Protokollierung erfolgt durch SAT in eine besonders geschützte systemglobale Datei (SAT-Protokolldatei) auf der BS2000-Kennung $SYSAUDIT.

Mit SATUT können nachträglich Datensätze aus der SAT-Protokolldatei ausgewählt und in eine Datei ausgegeben werden (Postselection). Die Auswahl der Protokollsätze kann über jedes protokollierte Feld erfolgen. Ein Datensatz wird nur übernommen, wenn die Datenfelder des zugehörigen Ereignisses bestimmte Regeln erfüllen. Für jeden Feldtyp kann eine der folgenden Bedingungen definiert werden:

– Der Wert des Feldes stimmt mit einem Element einer Liste überein.
– Der Wert des Feldes liegt innerhalb eines Intervalls.
– Das Datenfeld vom angegebenen Feldtyp ist im Datensatz vorhanden.

Weitere Informationen sind im BS2000/OSD-Handbuch „SECOS“ zu finden.

Der Aufbau der Protokollsätze ist im Anhang ab Seite 255 beschrieben.

11.5 Administration der SAT-Protokollierung

SAT protokolliert für UTM-Anwendungen ausgewählte sicherheitsrelevante UTM-Ereignisse. Bei der Generierung der Anwendung wird festgelegt, nach welchem Bearbeitungsresultat (Success, Failure) und nach welchen Kriterien (ereignisspezifisch, TAC-spezifisch oder benutzerspezifisch) die SAT-Protokollierung erfolgen soll. In diesem Kapitel sind die UTM-SAT-Administrationskommandos beschrieben, mit denen Sie die SAT-Protokollierung für Ihre UTM-Anwendung administrieren können.

Die UTM-SAT-Administrationskommandos sind jeweils eigene Transaktionscodes. Sie müssen daher bei der Generierung der Anwendung definiert werden. Die UTM-SAT-Administrationsfunktionen kann der UTM-SAT-Administrator nur über Dialog-TACs aufrufen. Die Namen der Transaktionscodes entnehmen Sie bitte der folgenden Tabelle:

Transaktionscode	**Administrationsfunktionen**
KDCMSAT	SAT-Protokollierung ein- und ausschalten SAT-Protokollierungswerte ändern Syntax des Kommandos abfragen
KDCISAT	Informationen über SAT-Protokollierungswerte anzeigen lassen Syntax des Kommandos abfragen

Sie können bei der Generierung festlegen, ob die SAT-Protokollierung automatisch bei jedem Start der Anwendung eingeschaltet werden soll oder nicht. Bei nicht eingeschalteter SAT-Protokollierung können trotzdem Kriterien für die SAT-Protokollierung generiert werden. Die Kriterien dienen dann als Vorbelegung für eine SAT-Protokollierung, die im laufenden Betrieb nach Bedarf ein- und ausgeschaltet werden kann.

Die generierten Protokollierungswerte können mit dem Administrationskommando KDCMSAT verändert werden. Die Änderung gilt für ereignisgesteuerte Protokollierung nur für die Dauer des aktuellen Anwendungslaufs. Für benutzer- und auftragsgesteuerte Protokollierung bleiben die Änderungen über das Anwendungsende hinaus erhalten.

Mit dem Administrationskommando KDCISAT können Sie sich die aktuell eingestellten Werte anzeigen lassen.

Die Eingaben der UTM-SAT-Administrationskommandos erfolgen im Line-Mode. Eingaben aus mit Editprofilen erstellten Ausgaben und formatierte Eingaben werden abgewiesen.

Die UTM-SAT-Administrationskommandos können nur von UTM-Benutzerkennungen aufgerufen werden, die UTM-SAT-Administrationsberechtigung haben. Die UTM-SAT-Administrationsberechtigung wird einer UTM-Benutzerkennung mit USER ...,PERMIT=SATADM oder PERMIT=(ADMIN,SATADM) bei der Generierung mit KDCDEF zugeordnet. Die UTM-Administrationsberechtigung (USER ..,PERMIT=ADMIN) allein berechtigt **nicht** zur UTM-SAT-Administration.

Damit Sie UTM-SAT-Administrationskommandos eingeben können, müssen bei der Generierung mit KDCDEF folgende Voraussetzungen erfüllt sein:

- Das Administrationsprogramm KDCSADM muss definiert sein (PROGRAM-Anweisung). Das Beispielprogramm KDCSADM gehört zum Lieferumfang von *open*UTM.
- Die UTM-SAT-Administrationskommandos KDCMSAT und KDCISAT müssen als Transaktionscodes definiert sein (TAC-Anweisung mit PROGRAM=KDCSADM und SATADM=Y).
- Zumindest eine Benutzerkennung muss mit UTM-SAT-Administrationsberechtigung generiert werden (USER-Anweisung mit PERMIT=SATADM). Die Administrationsberechtigung kann gleichzeitig auch an mehrere Benutzerkennung vergeben werden. Die UTM-SAT-Administrationsberechtigung ist an den Benutzer gekoppelt und nicht an ein Terminal, d.h. man kann von jedem Terminal aus administrieren. Soll nur von bestimmten Terminals aus administriert werden, so wird dies mit den normalen Funktionen des Zugriffsschutzes realisiert (Operanden KSET= und LOCK=).

Beispiel

Die Transaktionscodes für die Administration werden mit der Steueranweisung TAC, das Administrationsprogramm wird mit der Steueranweisung PROGRAM des Generierungstools KDCDEF definiert:

```
PROGRAM KDCSADM,COMP=ILCS
:
TAC KDCMSAT ,PROGRAM=KDCSADM,SATADM=Y
TAC KDCISAT ,PROGRAM=KDCSADM,SATADM=Y
:
```

Die Operanden PROGRAM=KDCSADM und SATADM=Y der TAC-Anweisungen können bei Voreinstellung mit DEFAULT entfallen:

```
DEFAULT TAC PROGRAM=KDCSADM,SATADM=Y
```

Hinweise

- Jeder Zugriff auf den TAC KDCMSAT (außer KDCMSAT HELP) wird protokolliert, auch bei ausgeschalteter SAT-Protokollierung.
- Die UTM-SAT-Administration über Asynchronaufträge ist **nicht** möglich.

11.6 UTM-SAT-Administrationskommandos

11.6.1 KDCISAT - Informationen über SAT-Protokollierungswerte abfragen

Mit dem Kommando KDCISAT kann sich der SAT-Administrator über die aktuell eingestellten Werte für die SAT-Protokollierung informieren oder die Syntax des Kommandos KDCISAT abfragen. KDCISAT liefert Informationen darüber, aus welchen UTM-Ereignisklassen die Ereignisse nach welchem Bearbeitungsergebnis (positiv oder negativ) protokolliert werden. Die Informationen können auch TAC- und/oder benutzerspezifisch abgefragt werden. Die UTM-Ereignisklassen sind bei der Beschreibung des Operanden EVENT bei der KDCMSAT-Anweisung aufgelistet (Seite 174ff).

Bei jeder Abfrage mit KDCISAT wird ausgegeben, ob die SAT-Protokollierung ein- oder ausgeschaltet ist.

```
KDCISAT␣      { [ TAC=tacname ] [, USER=username] | HELP }
```

KDCISAT ohne Operanden eingegeben:
: Es wird angezeigt, aus welchen Ereignisklassen Ereignisse protokolliert werden. Zu den einzelnen Ereignisklassen wird angegeben, ob erfolgreiche oder nicht erfolgreiche Ereignisse protokolliert werden (siehe Beispiel 1). Ausgegeben werden die Werte, die für alle Benutzer und für alle TACs gleich eingestellt sind. Das sind alle Werte der SAT-Protokollierung, die entweder in der SATSEL-Anweisung generiert oder mit dem Kommando KDCMSAT SATSEL=...,EVENT=(...) definiert wurden.

TAC=tac
: Es wird aus den einzelnen Ereignisklassen angezeigt, welche Ereignisse speziell für diesen TAC protokolliert werden (siehe Beispiel 2).

USER=user
: Es wird aus den einzelnen Ereignisklassen angezeigt, welche Ereignisse speziell für diesen Benutzer protokolliert werden.

TAC=tacname,USER=username
: Es wird für die einzelnen Ereignisklassen angezeigt, welche Ereignisse insgesamt für den angegebenen Benutzer und den angegebenen TAC protokolliert werden. Die Ausgabe setzt sich zusammen aus den eingestellten ereignisspezifischen Werten (KDCISAT ohne Operand), den TAC-spezifischen Werten (KDCISAT TAC=...) und den benutzerspezifischen Werten (KDCISAT USER=...) für die SAT-Protokollierung (siehe Beispiel 3).

HELP
: zeigt die Syntax dieses Kommandos an.

Ausgabe von KDCISAT

Alle Informationen werden auf das Terminal des SAT-Administrators ausgegeben. In der Spalte RESULT steht, ob Ereignisse protokolliert werden, die auf eine positive Ausführung oder auf eine negative Bearbeitung hinweisen etc.. Der in RESULT angegebene Wert ist das Ergebnis, das sich aus der Kombination der eingestellten SAT-Protokollierungsbedingungen ergibt. Das Ergebnis ist die Oder-Verknüpfung aus den für EVENT und gegebenenfalls USER und gegebenenfalls TAC eingestellten Protokollierungswerten (siehe Tabellen im Abschnitt „Preselection-Werte verknüpfen“ auf Seite 163).

Wird ein ungültiger USER oder TAC angegeben, so wird die Meldung `invalid TAC` oder `invalid USER` ausgegeben.

Beispiel 1

Bei der Eingabe von:

```
KDCISAT
```

erhalten Sie die folgende Ausgabe:

SAT	USER	TAC	EVENT [1]	RESULT
			SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH BOTH NONE NONE BOTH BOTH BOTH BOTH

[1] Die Bedeutung der Ereignisklassen (EVENT) und der Werte von RESULT sind beim Kommando KDCMSAT (Seite 174ff) beschrieben.

Beispiel 2

Bei der Eingabe von:

```
KDCISAT TAC=tac1
```

erhalten Sie die folgende Ausgabe:

SAT	USER	TAC	EVENT	RESULT
		tac1	SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH BOTH NONE NONE BOTH BOTH BOTH BOTH

Beispiel 3

Bei der Eingabe von:

```
KDCISAT TAC=tac2,USER=user2
```

erhalten Sie die folgende Ausgabe:

SAT	USER	TAC	EVENT	RESULT
ON	user2	tac2	SIGN CHANGE-PW START-PU END-PU GSSB TLS ULS ADM-CMD	BOTH BOTH NONE NONE BOTH BOTH BOTH BOTH

11.6.2 KDCMSAT - SAT-Protokollierung ändern

Bei der Generierung werden die Bedingungen für die SAT-Beweissicherung festgelegt. Es werden die Ereignisklassen angegeben, aus denen Ereignisse protokolliert werden sollen, und benutzerspezifisch, TAC-spezifisch oder ereignisspezifisch festgelegt, nach welchen Bearbeitungsergebnissen (erfolgreiche oder fehlerhafte Ausführung) protokolliert werden soll. Mit KDCMSAT kann der UTM-SAT-Administrator die generierten Werte modifizieren:

- Protokollierungswerte einstellen, jeweils mit einer eigenen Anweisung für EVENT, TAC oder USER.
- Protokollierung ein- oder ausschalten.

  Bei der Generierung können Sie SAT-Protokollierungswerte voreinstellen, ohne die SAT-Protokollierung selbst zu aktivieren. Mit KDCMSAT SAT=ON können Sie dann bei Bedarf die Protokollierung im laufenden Betrieb einschalten. Sie können die Protokollierung auch ohne Voreinstellungen bei der Generierung einschalten. Sie können dann mit KDCMSAT die Protokollierungswerte angeben.
- die Syntax dieses Kommandos abfragen.

Die Modifikationen für ereignisgesteuerte Protokollierung (SATSEL ...,EVENT=) und das Ein-/Ausschalten (SAT=ON/OFF) wirken nur bis zum Ende des Anwendungslaufs. Modifikationen für benutzer- und auftragsgesteuerte Protokollierung (SATSEL ...,TAC= und SATSEL...,USER=) bleiben über das Anwendungsende hinaus erhalten, auch bei UTM-F. Diese Werte können mit dem inversen KDCDEF in eine neue Generierung übernommen werden.

```
KDCMSAT␣      { SATSEL={ BOTH | SUCC | FAIL | NONE | OFF },
                       { EVENT=(event1, ..., eventn) |
                         TAC=(tacname1, ..., tacnamen) |
                         USER=(username1, ..., usernamen) }
               oder:
               SAT={ OFF | ON }
               oder:
               HELP }
```

SATSEL = steuert die Art der SAT-Protokollierung.

Der Wert von SATSEL ändert die generierte Protokollierungsart für das im folgenden Operanden angegebene Element der Anwendung.

Bei jedem Absetzen des Kommandos KDCMSAT können Sie jeweils die Einstellung für eines der Kriterien EVENT, TAC oder USER ändern.

BOTH Es werden sowohl erfolgreiche als auch nicht erfolgreiche Ereignisse protokolliert.

SUCC Erfolgreiche Ereignisse werden protokolliert.

FAIL Nicht erfolgreiche Ereignisse werden protokolliert.

NONE Keine EVENT-, TAC- oder USER-spezifische Ereignisauswahl.

OFF Diese Einstellung ist nur für das Kriterium EVENT möglich. Für die in EVENT angegebenen Ereignisklassen werden dann keine Ereignisse protokolliert, auch wenn die SAT-Protokollierung bei der Generierung im USER- oder TAC-Kommando eingeschaltet wurde.

EVENT=(event1,...,event*n*)
Gibt die Liste der Ereignisklassen an, für die die Bedingungen der SAT-Protokollierung verändert werden sollen. Die folgenden Ereignisklassen können angegeben und beliebig kombiniert werden. Es dürfen maximal 8 Werte angegeben werden.

SIGN Anmelden eines Benutzers

CHANGE-**P**W
Ändern des Passworts durch den Benutzer oder UTM-Administrator

START-PU
Starten eines Teilprogrammlaufs oder
Annahme eines Dialog- oder Asynchron-Auftrags

END-PU Ende eines Teilprogrammlaufs

GSSB Zugriff auf einen globalen sekundären Speicherbereich (GSSB)

TLS Zugriff auf einen terminalspezifischen Langzeitspeicher (TLS)

ULS Zugriff auf einen benutzerspezifischen Langzeitspeicher (ULS)

ADM-CMD
Ausführung eines Administrationsaufrufs

TAC=(tacname1, ..., tacname*n*)
Gibt die Liste der Transaktionscodes an, für die die Bedingungen der SAT-Protokollierung verändert werden sollen. Sie können eine Liste von maximal 10 Transaktionscodes angeben.

USER=(username1, ..., username*n*)
Gibt die Liste der Benutzer (USER) an, für die die Bedingungen der SAT-Protokollierung verändert werden sollen. Sie können eine Liste von maximal 10 Benutzernamen angeben.

SAT=ON/OFF SAT-Protokollierung ein-/ausschalten.

ON Die SAT-Protokollierung wird eingeschaltet.

Mit SAT=ON können Sie die SAT-Protokollierung bei Bedarf während des Anwendungslaufs einschalten für Werte, die bei der Generierung voreingestellt oder mit einem vorhergehenden KDCMSAT-Kommando eingestellt wurden.

OFF Die SAT-Protokollierung wird ausgeschaltet.

HELP zeigt die Syntax dieses Kommandos an.

Ausgabe von KDCMSAT

Am SAT-Administrator-Terminal werden die alten und neuen Werte der SAT-Protokollierung ausgegeben, die mit diesem Kommando verändert bzw. ein- oder ausgeschaltet wurden.

Beispiel 1

Bei der Eingabe von:

```
KDCMSAT SATSEL=SUCC,EVENT=(START-PU,END-PU)
```

erhalten Sie die folgende Ausgabe:

SAT	EVENT	SATSEL	
		NEW	OLD
ON	START-PU END-PU	SUCC SUCC	FAIL FAIL

Für die Kriterien TAC und USER wird ein entsprechendes Bild ausgegeben.

Beispiel 2

Bei der Eingabe von:

```
KDCMSAT SAT=ON
```

erhalten Sie die folgende Ausgabe:

SAT	NEW	OLD
	ON	OFF

12 Accounting

*open*UTM stellt Accounting-Funktionen zur Verfügung, die es dem Betreiber einer UTM-Anwendung ermöglichen, den Benutzern der UTM-Anwendung die in Anspruch genommenen Betriebsmittel zu verrechnen. Das UTM-Accounting nutzt die Möglichkeiten des Betriebssystems, um die Abrechnungsdaten zu ermitteln und die Daten in der Abrechnungsdatei des Betriebssystems zu erfassen.

Mit dem Abrechnungssystem RAV des BS2000 können diese Daten ausgewertet werden.

Detaillierte Informationen zum BS2000-Accounting finden Sie im BS2000/OSD-Handbuch „RAV (BS2000/OSD) - Rechenzentrums-Abrechnungsverfahren“.

Die Accounting-Funktionen, die das jeweilige Betriebssystem zur Verfügung stellt, können die Betriebsmittelauslastung und Leistung einer UTM-Anwendung nur als Ganzes erfassen. Wenn Sie die DV-Leistungen aber den einzelnen Benutzern der UTM-Anwendung zuordnen und in Rechnung stellen wollen, muss für das UTM-Accounting Folgendes berücksichtigt werden:

- Die Benutzer einer UTM-Anwendung werden durch Benutzerkennungen der UTM-Generierung repräsentiert und nicht durch Benutzerkennungen des Betriebssystems. Die von einem Benutzer in Anspruch genommenen Leistungen müssen also den einzelnen UTM-Benutzerkennungen zugeordnet werden können.
- In einer UTM-Anwendung ist eine Gruppe homogener Prozesse aktiv. Jeder Prozess bearbeitet nacheinander Aufträge für wechselnde Benutzer. Die innerhalb eines Prozesses in Anspruch genommenen Leistungen müssen deshalb pro aufgerufenen Service (d.h. für einzelne Teilprogrammläufe) ermittelt werden.
- Die Zeitbedingungen des OLTP-Betriebs erfordern eine Form der Leistungserfassung, die die Performance der Anwendung nicht beeinträchtigt.

Beim UTM-Accounting wird also der Verbrauch an Betriebsmitteln erfasst, der von den einzelnen Teilprogrammen beansprucht wird. Damit kann der Betriebsmittelverbrauch dem Transaktionscode (TAC) des jeweiligen Teilprogramms und somit auch dem UTM-Benutzer, der den zugehörigen Service gestartet hat, zugeordnet werden.

Über die vom UTM-Accounting erfassten Leistungen hinaus gibt es einen Grundbedarf an Betriebsmitteln, der beim Ablauf einer UTM-Anwendung anfällt, aber nicht direkt einem Benutzer zugeordnet werden kann. Das sind:

- Plattenbelegung für KDCFILE-, SYSLOG- und USLOG-Dateien sowie
- CPU-Verbrauch und I/Os für
  - Starten und Beenden der UTM-Prozesse
  - Verbindungsbehandlung der zugeordneten Clients
  - LPUT-Behandlung (Übertragung auf USLOG-Datei)
  - Aufbereitung von Druckerausgaben

Sollen diese Leistungen bei der Abrechnung berücksichtigt werden, dann müssen Sie sie den Benutzern als Pauschale in Rechnung stellen.

12.1 Begriffsdefinitionen

In diesem Abschnitt werden einige Begriffe genauer erläutert, die für das UTM-Accounting relevant sind.

Benutzer im Sinne des UTM-Accounting

Der Benutzer einer UTM-Anwendung, für den eine Abrechnung erstellt werden soll, wird i.A. durch die UTM-Benutzerkennung repräsentiert.

Bei Anwendungen oder Clients, die sich nicht explizit mit einer echten Benutzerkennung angemeldet haben, wird der Name der Verbindungs-Benutzerkennung (TS-Anwendungen und UPIC-Clients), der LU6-Sessionname (LU6-Partner) bzw. der OSI-Association-Name (OSI TP-Partner) verwendet.

In UTM-Anwendungen ohne Benutzerkennungen ordnet *open*UTM die Leistungen, die von Terminals, UPIC-Clients oder TS-Anwendungen in Anspruch genommenen wurden, ersatzweise den LTERM-Partnern zu.

Abrechnungsdatei

Alle Informationen, die das UTM-Accounting für die benutzerspezifische Abrechnung der Leistungen sammelt, schreibt *open*UTM in die Abrechnungsdatei des Betriebssystems.

Die Abrechnungsdatei wird vom BS2000-Systemadministrator verwaltet, der diese Abrechnungsdatei mit dem Tool RAV auswerten kann.

Betriebsmittel

Darunter werden folgende Leistungen zusammengefasst:

– technische DV-Leistungen, insbesondere CPU-Verbrauch und I/Os
– der Aufruf eines bestimmten Programms (Programmgebühr)

Kalkulationsphase

Die Kalkulationsphase dient als Orientierung für den Einsatz des Abrechnungsverfahrens.

In der Kalkulationsphase ermittelt *open*UTM für jedes aufgerufene Teilprogramm den Verbrauch der einzelnen Betriebsmittel und schreibt die Werte als Kalkulationssatz in die Abrechnungsdatei. Näheres ist in Abschnitt „Kalkulationsphase“ auf Seite 182 beschrieben.

Kalkulationssatz

Ein Kalkulationssatz ist ein Satz, den *open*UTM in der Kalkulationsphase pro Teilprogrammlauf in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMK.
Die Datenfelder des Kalkulationssatzes UTMK sind im Anhang auf Seite 254 beschrieben.

Gewicht

Pro Betriebsmittel können Sie ein Gewicht (Faktor) festlegen. Dieses Gewicht gibt an, wie das Betriebsmittel im Vergleich mit anderen Betriebsmitteln zu werten ist. Der Verbrauch eines Betriebsmittels geht dann als Produkt „Gewicht ∗ Betriebsmittelverbrauch“ in die Abrechnung ein. Die Gewichte für die einzelnen Betriebsmittel geben Sie bei der KDCDEF-Generierung in ACCOUNT an, siehe Abschnitt „Variante des Abrechnungsverfahrens festlegen“ auf Seite 184.

Abrechnungsphase

In der Abrechnungsphase ermittelt *open*UTM für jedes Teilprogramm den Betriebsmittelverbrauch. Nach Beendigung eines Teilprogrammlaufs ermittelt *open*UTM anhand der generierten Gewichte und des generierten Festpreises die Summe der Verbrauchswerte.

Folgenden Leistungen werden berücksichtigt:

– CPU-Verbrauch
– Ein-/Ausgaben auf Platte
– erzeugte Ausgabeaufträge für Drucker
– Festpreis für den Aufruf eines Teilprogramms

Das Ergebnis ist eine Anzahl von Verrechnungseinheiten, die auf den benutzerspezifischen Verrechnungseinheitenzähler aufaddiert wird.

*open*UTM schreibt erst dann einen Satz mit dem Inhalt dieses Zählers in die Abrechnungsdatei,
– wenn der Benutzer sich abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist,
– oder wenn die Anwendung normal beendet wird
– oder wenn ein bestimmter Maximalwert überschritten wird. Diesen Maximalwert legen Sie bei der KDCDEF-Generierung mit ACCOUNT ...,MAXUNIT= fest.

Gewichte und Festpreise müssen Sie vor dem Start der Abrechnungsphase in die Generierung der Anwendung einbringen. Dabei können Sie wählen zwischen
– Festpreis-Abrechnung,
– verbrauchsorientierter Abrechnung,
– und einer Kombination der beiden Varianten.

Eine detaillierte Beschreibung der Abrechnungsphase finden Sie in Abschnitt „Abrechnungsphase“ auf Seite 186.

Die Abrechnungsphase des UTM-Accounting kann im laufenden Betrieb der UTM-Anwendung ein- und ausgeschaltet werden.

Abrechnungssatz

Ein Abrechnungssatz ist ein Satz, den *open*UTM in der Abrechnungsphase in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMA.
Die Datenfelder des Abrechnungssatz UTMA sind im Anhang auf Seite 253 beschrieben.

Verrechnungseinheiten

Verrechnungseinheiten sind das Produkt aus Verbrauch und Gewicht des jeweiligen Betriebsmittels. In der UTM-Abrechnung werden nur Verrechnungseinheiten gezählt. Mit Hilfe von RAV können sie in Kosten umgerechnet werden, die den Benutzern in Rechnung gestellt werden.

Verrechnungseinheitenzähler

*open*UTM führt in einer UTM-Anwendung pro Benutzer einen Verrechnungseinheitenzähler und akkumuliert dort den Verbrauch der Verrechnungseinheiten pro Benutzer.

Festpreis-Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten in Rechnung gestellt. Diese Anzahl wird bei der Anwendungsgenerierung dem Transaktionscode zugeordnet. Die Gewichte der anderen Betriebsmittel sind dabei Null. Dabei können Sie einzelne Services auch kostenlos anbieten, z.B. Auskunftsfunktionen.

Verbrauchsorientierte Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell ermittelt wird. Die Verbrauchswerte für die Betriebsmittel werden entsprechend der generierten Gewichte gewichtet. Für den Aufruf eines Teilprogramms wird kein Festpreis berechnet.

Rechenzentrum-Abrechnungs-Verfahren (RAV)

Mit RAV können die Sätze weiterverarbeitet werden, die von UTM-Anwendungen in die BS2000-Abrechnungsdatei geschrieben werden. Dafür steht im RAV die Komponente RAV-UTM zur Verfügung.

12.2 Phasen des Accounting

Für die Durchführung des Accounting in UTM-Anwendungen sind folgende Schritte erforderlich:

– Kalkulationsphase
– Abrechnungsverfahren festlegen
– Abrechnungsphase
– Auswertung

12.2.1 Kalkulationsphase

Die Kalkulationsphase liefert Orientierungswerte, mit deren Hilfe Sie die einzelnen Betriebsmittel gewichten und Festpreise für die Inanspruchnahme eines Services festlegen können. Dabei ermittelt *open*UTM für jeden Teilprogrammlauf den Betriebsmittelverbrauch, stellt am Programmlaufende einen Kalkulationssatz vom Satztyp UTMK zusammen und schreibt ihn in die Abrechnungsdatei.

Die Kalkulationsphase kann im laufenden Betrieb jederzeit durch die UTM-Administration ein- und ausgeschaltet werden, z.B. um die generierten Gewichte zu überprüfen und gegebenenfalls bei der Neugenerierung zu aktualisieren.

Sie sollten jedoch beachten, dass *open*UTM bei eingeschalteter Kalkulationsphase nach jedem Teilprogrammlauf einen Satz in die Abrechnungsdatei schreibt. Dadurch wird die Performance der Anwendung belastet.

Kalkulationsphase einschalten

Die Kalkulationsphase können Sie bei der KDCDEF-Generierung oder per Administration einschalten, siehe auch *open*UTM-Handbuch „Anwendungen generieren“ und *open*UTM-Handbuch „Anwendungen administrieren”:

- per KDCDEF-Anweisung ACCOUNT ACC=CALC
- oder per UTM-Administration
  – durch das Kommando KDCAPPL CALC=ON
  – oder über WinAdmin
  – oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Im BS2000-Accounting muss der Systemverwalter den Satztyp UTMK einschalten.

Kalkulationsphase ausschalten

Die Kalkulationsphase können Sie nur per UTM-Administration ausschalten:

- durch das Kommando KDCAPPL CALC=OFF
- oder über WinAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten eines Kalkulationssatzes

Ein Kalkulationssatz enthält folgende Daten:

- Zeitstempel des BS2000-Accounting
- Name der UTM-Anwendung
- Transaktionscode (TAC) des Teilprogramms
- CPU-Verbrauch im UTM-Prozess in msec
- CPU-Verbrauch im DB-System in msec, sofern das verwendete DB-System entsprechende Daten an *open*UTM liefert.
- Anzahl I/Os im UTM-Prozess
- Anzahl I/Os im DB-System
- Länge der Eingabenachricht in byte
- Länge der Ausgabenachricht in byte
- Anzahl Ausgabeaufträge an Drucker
- Verrechnungseinheiten für LTAC-Aufrufe
- UTM-Benutzer, der den Service aufgerufen hat
- Name des LTERM-Partners, über den der Benutzer angemeldet ist
- Realzeit des Teilprogrammlaufs (msec)

Bei den Ausgabenachrichten werden auch die mitgerechnet, die an ein Folgeteilprogramm gerichtet sind (z.B. nach PEND PR).

Aus den Kalkulationssätzen kann mit RAV eine Auswertung erzeugt werden, die den mittleren Betriebsmittelverbrauch pro TAC anzeigt. Sind mehrere UTM-Anwendungen im Einsatz, wird für jede UTM-Anwendung eine Auswertung erzeugt.

12.2.2 Variante des Abrechnungsverfahrens festlegen

Zunächst müssen Sie festlegen, ob Sie über Festpreis, über den Verbrauch oder über eine Kombination aus diesen beiden Varianten abrechnen. Ihre Entscheidung hängt davon ab, ob Sie bestimmte Leistungen der Anwendung mit festen Preisen anbieten oder den tatsächlichen Betriebsmittelverbrauch in Rechnung stellen wollen.

Festpreis-Abrechnung

Bei der Festpreis-Abrechnung kostet ein Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten. Orientierungswerte dafür liefert die Kalkulationsphase. Daher ist eine Festpreis-Abrechnung die einfachste Lösung.

Die Anzahl der Verrechnungseinheiten geben Sie bei der KDCDEF-Generierung in der TAC-Anweisung beim Operanden TACUNIT an, siehe *open*UTM-Handbuch „Anwendungen generieren“.

```
TAC tacname,PROGRAM=progname,TACUNIT=anzahl_verrechnungseinheiten
```

Für jeden vom Benutzer aufgerufenen Transaktionscode wird der in TACUNIT angegebene Wert auf den benutzerspezifischen Verrechnungseinheitenzähler aufaddiert.

Bei der Festpreis-Abrechnung können Sie einige Services (z.B. Auskunftsfunktionen) auch kostenfrei zur Verfügung stellen. Dazu müssen Sie die zugehörigen Transaktionscodes wie folgt generieren:

```
TAC ... TACUNIT=0
```

Bei Verteilter Verarbeitung gilt Entsprechendes für die Anweisung LTAC und den Operanden LTACUNIT, siehe Abschnitt „Abrechnung bei verteilter Verarbeitung“ auf Seite 189.

Bei einer Festpreis-Abrechnung müssen Sie die Gewichte der Betriebsmittel in der KDCDEF-Anweisung ACCOUNT auf 0 setzen (=Standardwert).

Verbrauchsorientierte Abrechnung

Bei dieser Variante wird dem Benutzer der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell in der Abrechnungsphase ermittelt wird. Für die einzelnen Betriebsmittel müssen Sie Gewichte festlegen. Ein Gewicht ist ein Faktor, mit dem die verbrauchten Einheiten multipliziert werden. Bei der Wahl dieser Gewichte können Sie sich an den Verbrauchsdaten orientieren, die Sie in der Kalkulationsphase ermittelt haben.

Die Gewichte werden anwendungsspezifisch in der KDCDEF-Anweisung ACCOUNT festgelegt, d.h. sie gelten für alle Teilprogrammläufe.

Die Festlegung der Gewichte ist zwangsläufig subjektiv und hängt von der Installationsumgebung ab. Folgende Betriebsmittel können gewichtet werden:

– CPU-Verbrauch (ACCOUNT-Operand CPUUNIT)
– Ein-/Ausgaben auf Hintergrundspeicher (ACCOUNT-Operand IOUNIT)
– Druckerausgaben (ACCOUNT-Operand OUTUNIT)

Näheres siehe *open*UTM-Handbuch „Anwendungen generieren“.

Beispiel für die Generierung dieser Variante

```
ACCOUNT ACC=ON,CPUUNIT=15,IOUNIT=5,OUTUNIT=20
TAC tacname,PROGRAM=progname,TACUNIT=0
TAC ....
```

Pro Aufruf eines Transaktionscodes wird dann die folgende Summe auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

15 ∗ CPU-Verbrauch + 5 ∗ I/O-Verbrauch + 20 ∗ Druckausgaben-Verbrauch

Kombination aus Festpreis- und verbrauchsorientierter Abrechnung

Sie können für Ihre Abrechnung die beiden obigen Varianten auch kombinieren, indem Sie für den Aufruf eines Transaktionscodes einen bestimmten Festpreis festlegen und zusätzlich den Verbrauch der Betriebsmittel (z.B. den CPU-Verbrauch) verrechnen.

In der Abrechnungsphase wird beim Aufruf eines Transaktionscodes die folgende Summe gebildet und auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

TACUNIT (Festpreis für den Aufruf des Teilprogramms)
+ CPUUNIT ∗ CPU-Verbrauch + IOUNIT∗ I/O-Verbrauch
+ OUTUNIT ∗ Druckausgaben-Verbrauch

Beispiel für die Generierung dieser Variante

```
ACCOUNT ACC=ON,CPUUNIT=15
TAC tacnam1,PROGRAM=progname1,TACUNIT=1
TAC tacnam2,PROGRAM=progname2,TACUNIT=2
:
:
```

12.2.3 Abrechnungsphase

In der Abrechnungsphase ermittelt *open*UTM die pro Teilprogrammlauf verbrauchten Betriebsmittel, berechnet daraus und aus den generierten Gewichten und Festpreisen eine gewichtete Summe. Diese Summe addiert *open*UTM auf den Verrechnungseinheitenzähler des UTM-Benutzers. Der Wert dieses Zählers ist im Abrechnungssatz enthalten, den *open*UTM in die Abrechnungsdatei schreibt.

*open*UTM schreibt immer dann einen Abrechnungssatz, wenn für den Benutzer eine bestimmte Anzahl von Verrechnungseinheiten aufaddiert sind oder wenn sich der Benutzer abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist. Die Anzahl von Verrechnungseinheiten, bei der *open*UTM einen Abrechnungssatz schreibt, legen Sie bei der KDCDEF-Generierung in ACCOUNT MAXUNIT= fest. Dabei müssen Sie Folgendes beachten:

- Den Wert von MAXUNIT sollten Sie nicht zu klein wählen, da ein zu häufiges Schreiben von Abrechnungssätzen die Performance der Anwendung beeinträchtigen könnte.
- Den Wert von MAXUNIT sollten Sie nicht zu groß wählen, da bei einem Abbruch der Anwendung die Verrechnungseinheiten, die noch nicht in die Abrechnungsdatei geschrieben wurden, verloren gehen können (die Abrechnung unterliegt nicht der Transaktionssicherung).

Nachdem der Abrechnungssatz in die Abrechnungsdatei geschrieben wurde, werden der Verrechnungseinheitenzähler und der Zähler für die Anzahl der aufgerufenen TACs auf Null gesetzt.

Abrechnungsphase einschalten

Mit der KDCDEF-Steueranweisung ACCOUNT ACC=ON wird bei der Generierung die Abrechnung für die UTM-Anwendung eingeschaltet.

Zusätzlich kann die Abrechnungsphase auch im laufenden Betrieb durch die UTM-Administration ein- und ausgeschaltet werden

– durch das Kommando KDCAPPL ACCOUNT=ON
– oder durch WinAdmin
– oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Im BS2000-Accounting muss der Systemverwalter den Satztyp UTMA einschalten.

Abrechnungsphase ausschalten

Die Abrechnungsphase können Sie nur per Administration ausschalten:

– durch das Kommando KDCAPPL ACCOUNT=OFF
– oder über WinAdmin
– oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten des Abrechnungssatzes

Der Abrechnungssatz ist vom Satztyp UTMA. Er enthält folgende Daten:

– Zeitstempel des BS2000-Accounting
– Name der UTM-Anwendung
– UTM-Benutzerkennung
– Zeitpunkt der Anmeldung des Benutzers auf der aktuellen Verbindung
– Wert des Verrechnungseinheitenzählers
– Anzahl der aufgerufenen TACs mit TACUNIT > 0 seit der Anmeldung oder seitdem der letzte Satz geschrieben wurde

Sie können Kalkulationsdaten auch bei laufender Abrechnungsphase erfassen. Damit können Sie die Gewichte jederzeit überprüfen.

12.2.4 Auswertung

Das Ergebnis der Abrechnungsphase sind die Abrechnungssätze in der Abrechnungsdatei des BS2000. Sie können mit RAV ausgewertet werden. Die hierfür notwendigen Programme sind Bestandteil von RAV und werden zusammen mit diesem Produkt ausgeliefert. Sehen Sie hierzu auch Benutzerhandbuch „RAV (BS2000/OSD) - Rechenzentrums-Abrechnungsverfahren“.

Der Aufbau der UTM-Abrechnungssätze ist im Anhang auf Seite 253 beschrieben.

12.2.5 Fehlersituationen

Kann das BS2000-Accounting auf Grund eines Fehlers einen Abrechnungssatz und/oder einen Kalkulationssatz nicht schreiben, z.B. weil nicht genügend Platz auf der Platte ist, erzeugt *open*UTM die Meldung K079 und beendet die Kalkulations- und/oder Abrechnungsphase. Ein Insert der Meldung K079 gibt die Ursache des Fehlers an. Die Anwendung läuft weiter.

Nach Behebung des Fehlers kann die Kalkulations- und/oder Abrechnungsphase durch die UTM-Administration (z.B. mit dem Administrationskommando KDCAPPL) wieder eingeschaltet werden.

12.3 Abrechnung bei verteilter Verarbeitung

Die lokale Anwendung (Auftraggeber) nutzt Leistungen, die Teilprogramme in fernen Partneranwendungen (Auftragnehmer) für sie erbringen. In diesem Fall kann in der Auftraggeber-Anwendung der dabei anfallende Betriebsmittelverbrauch als Festpreis in Rechnung gestellt werden. Dazu ordnen Sie den LTACs in der Auftraggeber-Anwendung Verrechnungseinheiten als Festpreise zu. LTACs sind die Transaktionscodes, die in der Auftraggeber-Anwendung für einen Vorgang in einer Auftragnehmer-Anwendung definiert werden.

Näheres siehe *open*UTM-Handbuch „Anwendungen generieren“, Anweisung LTAC, Operand LTACUNIT.

Kalkulationsphase (Festlegen der Festpreise)

In der **Auftragnehmer-Anwendung** wird in der Kalkulationsphase der mittlere Betriebsmittelverbrauch der Teilprogramme ermittelt, die Services für die Auftraggeber-Anwendung erbringen. Anhand der hier ermittelten Verbrauchswerte können Sie die Festpreise festlegen, die den Benutzern der LTACs in der Auftraggeber-Anwendung berechnet werden sollen.

In der **Auftraggeber-Anwendung** zählt *open*UTM in einem Feld des Kalkulationssatzes die Verrechnungseinheiten, die bei den Aufrufen von LTACs anfallen.

Abrechnungsphase

In der **Auftragnehmer-Anwendung** werden alle Verbrauchswerte, die bei der Bearbeitung von Aufträgen für eine Auftraggeber-Anwendung anfallen, wie folgt zugeordnet:

– Bei LU6.1 den Sessions (LSES) zum Auftraggeber

– Bei OSI TP den Associations (OSI-LPAP ... ‚ASSOCIATION-NAME=), falls sich der OSI-TP-Auftraggeber nicht unter einer echten Benutzerkennung angemeldet hat.

Die erbrachten Leistungen werden also der Auftraggeber-Anwendung insgesamt in Rechnung gestellt. Die Aufwände für die einzelnen Benutzer der Auftraggeber-Anwendung können nicht ermittelt werden.

In der **Auftraggeber-Anwendung** addiert *open*UTM beim Aufruf eines LTACs die Verrechnungseinheiten auf den Verrechungseinheitenzähler des Benutzers auf, die in der KDCDEF-Generierung in der LTAC-Anweisung angegeben wurden.

12.4 Einschränkungen

Bei der Nutzung des UTM-Accounting ist Folgendes zu beachten:

- Das Schreiben von Abrechnungsinformationen unterliegt nicht der Transaktionssicherung, deshalb können beim Abbruch einer Anwendung Verrechnungseinheiten verloren gehen. Der Maximalwert pro Benutzer kann per Generierung begrenzt werden.
- Für Anwendungen mit verteilter Verarbeitung wird in der Kalkulationsphase jeder LTAC-Aufruf mitgezählt. Es wird nicht berücksichtigt, ob nach der PEND-Verarbeitung eine Session belegt werden konnte oder nicht.
- Die Erfassung des Betriebsmittelverbrauchs beginnt vor dem Start eines Teilprogramms, sie endet in der Verarbeitung des PEND-Aufrufs. Die übrige Verarbeitungsleistung (Grundverbrauch) der UTM-Tasks wird den Benutzern nicht in Rechnung gestellt.
- Das Rücksetzen einer Transaktion hat folgende Auswirkungen: Alle Werte bis auf CPU und I/Os werden zurückgesetzt. Da *open*UTM die Verbrauchswerte in der PEND-Verarbeitung aufsummiert, kann eine Rücksetzaktion Verbrauchswerte nur dann zurücksetzen, wenn sie im aktuellen Teilprogrammlauf entstehen.
- Sind seit dem letzten Start der Anwendung für den Benutzer nur Asynchronaufträge verarbeitet worden, steht im Abrechnungssatz als Zeitpunkt der Anmeldung an die Anwendung der Wert Null.
- Für den Event-Exit VORGANG wird der Betriebsmittelverbrauch nur am Anfang des Vorgangs erfasst.
- Für den Event-Service BADTACS kann in der Abrechnungsphase kein Teilprogramm-Gewicht berücksichtigt werden.

13 Leistungskontrolle mit SM2 und KDCMON

Die Performance einer UTM-Anwendung wird von verschiedenen Faktoren beeinflusst. Die Einflussfaktoren liegen zum einen im System-Umfeld einer UTM-Anwendung (Ausbau des Arbeitsspeichers, Leistungsfähigkeit der Peripherie) und zum anderen in der UTM-Anwendung selbst (Konfiguration der Anwendung und Aufbau der Teilprogramme). Im laufenden Betrieb einer Anwendung sollten Sie regelmäßig Leistungskontrollen durchführen, damit Sie rechtzeitig Hinweise auf Leistungsengpässe erhalten. Für UTM-Anwendungen stehen Ihnen die folgenden Instrumente zur Leistungskontrolle zur Verfügung:

- BS2000-Software Monitor SM2
- UTM-Messmonitor KDCMON mit dem Auswertungstool KDCEVAL
- Informationsservices der UTM-Administration

Zusätzlich können Sie die Instrumente der Datenbanksysteme nutzen, z.B. SESCOS-Trace für SESAM/SQL.

Software Monitor SM2

Der Software Monitor SM2 des BS2000/OSD liefert statistische Daten über die Leistungen und die Auslastung der Betriebsmittel. Zusammen mit dem Subsystem UTM-SM2 ist es möglich, auch anwendungsspezifische Daten zu ermitteln und anzuzeigen. Diese Funktionen sollten Sie im laufenden Betrieb nutzen, um das Verhalten einer UTM-Anwendung zu überwachen und um Leistungsengpässe aufzudecken. Die Auswertung der Messdaten durch SM2 erlauben jedoch keine Aussage über einzelne Objekte der UTM-Anwendung, z.B. Teilprogramme. Sie zeigt vielmehr das Verhalten der gesamten Anwendung auf, z.B. Mittelwerte zu Transaktionsrate, Durchsatz und Bearbeitungszeit. Siehe Abschnitt „Auswertungslisten“ auf Seite 206.

UTM-Messmonitor KDCMON

Für UTM-Anwender steht der UTM-Messmonitor KDCMON zur Verfügung. KDCMON zeichnet Informationen über den Ablauf von UTM-Anwendungen und Anwenderteilprogrammen auf. Zeichnen sich Leistungsengpässe ab, können Sie mit KDCMON Daten erfassen. Die gesammelten Daten müssen Sie mit dem Tool KDCEVAL auswerten. Anhand dieser Auswertung können Sie eine detaillierte Analyse durchführen. Siehe ab Seite 206.

KDCMON ist damit ein wichtiges Werkzeug zur Beurteilung der Performance in einer UTM-Anwendung. KDCMON kann z.B. zur detaillierten Leistungsuntersuchung eingesetzt werden, wenn Messungen mit SM2 bzw. mit Hilfe der UTM-Administration auf Leistungsengpässe hinweisen.

Informationsservices der UTM-Administration

Detaillierte Informationen zur Diagnose von Leistungsengpässen können Sie auch per UTM-Administration abfragen, z.B. über das Administrationskommando KDCINF oder über das Administrationsprogramm WinAdmin. Das Kommando KDCINF STATISTICS liefert Daten über die Auslastung einzelner ausgewählter Komponenten Ihrer UTM-Anwendung, wie beispielsweise der Clients. Mit dem Kommando KDCINF STATISTICS können Sie gleichzeitig allgemeine Statistikinformationen über die Auslastung der gesamten Anwendung abfragen und Kenngrößen für die Leistungskontrolle und die Beurteilung der Performance Ihrer UTM-Anwendungen im laufenden Betrieb ermitteln, wie beispielsweise Belegung des Pagepools, Anzahl der aktuell angemeldeten Benutzer, Anzahl der durchgeführten Dialog- oder Asynchron-Transaktionen pro Sekunde, offene Dialog- und Asynchronvorgänge usw.. Ob die UTM-Anwendung Daten an SM2 liefert, können Sie mit dem Administrationskommando KDCINF SYSPARM abfragen. Sehen Sie dazu im *open*UTM-Handbuch „Anwendungen administrieren“.

Wenn Sie die UTM-Anwendung mit dem grafischen Administrationsarbeitsplatz WinAdmin administrieren, können die Statistikdaten auch grafisch dargestellt werden.

13.1 Messdatenerfassung mit SM2

Der Software-Monitor SM2 in BS2000/OSD erfasst statistische Daten über Leistung und Auslastung der Betriebsmittel.

*open*UTM kann auf Anforderung von SM2 Daten über UTM-Anwendungen an SM2 liefern, die für eine erste Beurteilung der Performance wichtig sind. Die Menge der Daten ist unabhängig von der Größe der Konfiguration. Die Daten zeigen das Verhalten der gesamten Anwendung auf.

Zur Auswertung der von SM2 erfassten Messdaten dient die SM2-Komponente SM2R1, sowie die SM2-Komponente SM2-PA zur Auswertung benutzerspezifischer Messwertdateien. SM2 erfasst Messwerte und gibt sie auf Anforderung direkt am Terminal für eine Echtzeitüberwachung (online) aus. Zusätzlich sammelt SM2 die Messdaten in einer Datei und die gespeicherten Daten können zu einem späteren Zeitpunkt ausgewertet werden (offline). In diesem Fall können eigene Benutzerprogramme die Daten zeitversetzt auswerten. SM2R1 wertet die Messwerte aus, die SM2 erfasst und in eine systemglobale Messwertdatei schreibt.

Für die Leistungskontrolle einer UTM-Anwendung stehen Ihnen auch andere Funktionen von SM2 zur Verfügung, z.B. globale Prozess-Auswertung, oder Vermessung der Prozesse einer UTM-Anwendung und Programmanalyse durch SM2-PA.

Voraussetzungen für die UTM-Messdatenerfassung durch SM2

Damit *open*UTM Daten an SM2 liefern kann und SM2 UTM-Daten sammeln, speichern und aufbereiten kann, müssen die im Folgenden beschriebenen Voraussetzungen erfüllt sein.

- Das Subsystem UTM-SM2 muss installiert und geladen sein.

  Für die Datenlieferung von *open*UTM an SM2 ist das Subsystem UTM-SM2 notwendig. UTM-SM2 dient als Kommunikationsbaustein zwischen den Prozessen der UTM-Anwendung und SM2. Es ist als eigenes Subsystem realisiert und ist in BS2000-GA (Grundausbau) enthalten. Wie das Subsystem UTM-SM2 vom BS2000-Systemverwalter zu installieren ist, ist im Anhang auf Seite 238 beschrieben.

  UTM-SM2 kann wie folgt geladen werden:

  1. Durch den BS2000-Systemverwalter mit dem Kommando:
     `/START-SUBSYSTEM SUBSYSTEM-NAME=UTM-SM2`
  2. Automatisch beim Start der Anwendung, wenn diese mit MAX SM2=ON generiert ist.
  3. Beim Einschalten der Datenlieferung durch die Administration, siehe unten. Dazu muss die Anwendung mit MAX SM2=OFF oder MAX SM2=ON generiert sein.

Bei Bedarf kann das Subsystem UTM-SM2 mit dem Kommando STOP-SS auch im laufenden Betrieb entladen werden. Das kann z.B. zum Austausch des Subsystems erforderlich werden. SM2 und die UTM-Anwendungen beenden dann ihre Zusammenarbeit mit UTM-SM2.

Nach dem Neuladen muss die Datenlieferung wieder explizit für jede UTM-Anwendung eingeschaltet werden.

- Die Datenlieferung von *open*UTM an SM2 muss in der UTM-Anwendung generiert sein.

  Damit *open*UTM Daten über eine UTM-Anwendung an SM2 liefern kann, muss dies bei der Generierung der Anwendung angegeben werden. Dazu geben Sie in der MAX-Anweisung im Operanden SM2 den Wert ON oder OFF an.

  Wird MAX...,SM2=ON angegeben, so wird die Datenlieferung an SM2 beim Start der Anwendung eingeschaltet. Sie kann dann im laufenden Betrieb bei Bedarf per UTM-Administration aus- und wieder eingeschaltet werden.

  Wird MAX...,SM2=OFF angegeben, so ist die Datenlieferung an SM2 für diese Anwendung erlaubt. Sie muss aber im laufenden Betrieb durch die UTM-Administration explizit eingeschaltet werden.

  Wenn MAX ...,SM2=NO generiert wird, so darf *open*UTM für diese Anwendung keine Daten an SM2 liefern. Die Datenlieferung kann dann auch nicht von der UTM-Administration veranlasst werden.

- Einschalten der Datenlieferung an SM2 durch die UTM-Administration.

  Mit dem Kommando KDCAPPL SM2=ON kann der UTM-Administrator die Datenlieferung an SM2 einschalten, wenn dies in der Generierung vorgesehen wurde. Mit KDCAPPL SM2=OFF wird die Datenlieferung ausgeschaltet.

  Mit dem Kommando KDCINF SYSPARM kann der UTM-Administrator feststellen, ob die Anwendung Daten an SM2 liefern darf und ob sie derzeit Daten liefert.

  Das Ein- und Ausschalten der Datenlieferung an SM2 ist auch über die „Programmierschnittstelle Administration“ oder über das Administrationsprogramm WinAdmin möglich.

- Der SM2-Verwalter (privilegierter SM2-Benutzer) muss das Sammeln von UTM-Daten durch SM2 starten.

  Der SM2-Verwalter muss mit dem Kommando

  ```
  START-MEASUREMENT-PROGRAM TYPE=UTM
  ```

  veranlassen, dass SM2 Daten zu UTM-Anwendungen sammelt und auswertet. Zum Ausschalten gibt es analog dazu das STOP-MEASUREMENT-PROGRAM-Kommando.

Ist das Subsystem UTM-SM2 geladen, so kann von SM2-Seite das Sammeln von UTM-Daten jederzeit eingeschaltet werden, unabhängig davon, ob von UTM-Seite die Datenlieferung eingeschaltet ist oder nicht. SM2 kann mit der Verarbeitung der Daten jedoch erst beginnen, wenn in der jeweiligen UTM-Anwendung die Datenlieferung eingeschaltet wird.

Ausgabe und Auswertung der Messdaten

Der SM2-Benutzer kann sich die von *open*UTM gelieferten Daten auf einem SM2-Bildschirm (UTM-Report bzw. UTM-Application-Report) online anzeigen lassen. SM2 speichert die Daten auch in einer systemglobalen Messwertdatei. Sie können sich die Daten somit auch später von SM2R1 auswerten lassen.

Der SM2-Bildschirm "UTM-REPORT" enthält eine tabellarische Übersicht von Daten zu allen UTM-Anwendungen, die derzeit Daten an SM2 liefern. Es wird eine Zeile pro UTM-Anwendung ausgegeben.

Eine periodische Ausgabe des SM2-Bildschirms "UTM-REPORT" erhalten Sie mit dem Kommando

```
REPORT UTM
```

Aktuelle Messwerte zu einer oder mehreren ausgewählten UTM-Anwendungen liefert der SM2-Bildschirm "UTM-APPLICATION-REPORT".

Eine periodische Ausgabe des SM2-Bildschirms "UTM-APPLICATION-REPORT" erhalten Sie mit dem Kommando

```
SELECT-UTM-APPLICATION (application1,application2,...)
```

Für *application1*,*application2*,... geben Sie die Namen der UTM-Anwendungen an, deren Verhalten Sie online verfolgen wollen. Sie müssen für *application1,..* jeweils den in der MAX-Anweisung generierten Namen der Anwendung angeben.

Die Bedeutung der ausgegebenen Daten ist im Benutzerhandbuch „SM2“ erklärt, die Begriffe halten sich an den Sprachgebrauch von *open*UTM.

Neben der Möglichkeit, die Messdaten online zu verfolgen, gibt es folgende Auswertungen mit SM2R1 von Daten aus UTM-Anwendungen:

– die Reports 128 bis 133
– SUMMARY UTM

Sehen Sie dazu das Benutzerhandbuch „SM2“.

13.2 UTM-Messmonitor KDCMON

KDCMON ist eine eingeschränkte Variante von COSMOS. COSMOS ist ein Werkzeug, das in BS2000/OSD zur Leistungskontrolle eingesetzt wird. Mit KDCMON werden nur UTM- und bestimmte DB-Ereignisse aufgezeichnet. KDCMON und COSMOS können zu gleicher Zeit an einer Anlage ablaufen, ebenso können SM2 und KDCMON zusammen eingesetzt werden.

KDCMON kann im laufenden Betrieb eingeschaltet und nach der gewünschten Messdauer wieder abgeschaltet werden. Die Daten können auf Band oder Platte geschrieben werden. Bei größeren Datenmengen sollten die Daten auf Bändern erfasst werden; dadurch können Stausituationen bei der Datenerfassung vermieden werden.

Zur Auswertung der von KDCMON erfassten Daten stehen die Tools KDCPMSM und KDCEVAL zur Verfügung:

– KDCPMSM setzt die von KDCMON erfassten Daten um und sortiert sie
– KDCEVAL erzeugt aus den umgesetzten Daten Auswertungslisten

KDCMON ist auch im Parallelbetrieb von UTM-Versionen einsetzbar, d.h. KDCMON kann Daten von UTM-Anwendungen aufzeichnen, die unter verschiedenen UTM-Versionen im gleichen BS2000-System ablaufen.

13.2.1 Erfassung starten und beenden

Die Erfassung der Daten wird in zwei Schritten gestartet:

- zuerst müssen Sie KDCMON starten
- danach schalten Sie die Datenaufzeichnung für die UTM-Anwendungen ein, die überprüft werden

KDCMON ist ein eigenständiges Subsystem in BS2000/OSD und ist in BS2000-GA (Grundausbau) enthalten. KDCMON muss vom Systemverwalter installiert und geladen werden. Sehen Sie dazu auch Abschnitt „Subsystem KDCMON“ auf Seite 240.

KDCMON starten

Vor dem Start von KDCMON müssen Sie zunächst eine Datei einrichten, in die KDCMON die erfassten Daten schreiben soll. Dazu müssen Sie folgende Kommandos absetzen:

- für die Erfassung auf Platten-Dateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile          -
/     ,ACCESS-METHOD=UPAM

/MODIFY-FILE-ATTRIBUTES FILE-NAME=kdcmonfile                  -
/     ,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE                   -
/      (PRIMARY-ALLOCATION=xxx,SECONDARY-ALLOCATION=yyy))
```

  xxx und *yyy* müssen Vielfache von 12 sein (sonst DVS-Fehler).

- für die Erfassung auf Band-Dateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile          -
/     ,ACCESS-METHOD=BTAM

/MODIFY-FILE-ATTRIBUTES FILE-NAME=kdcmonfile                  -
/     ,SUPPORT=*TAPE(VOLUME=xxxxxx,DEVICE-TYPE=TAPE-C4)
```



Blockgröße und die Satzlänge dürfen für die Datei nicht angegeben werden.

Danach können Sie KDCMON unter der Kennung $TSOS starten. Das Programm zum Starten von KDCMON wird in der Datei SYSPRG.KDCMON.*nnn* bereitgestellt:

```
/START-PROGRAM FROM-FILE=$userid.SYSPRG.KDCMON.nnn
```

Das Versionskennzeichen *nnn* steht für die BS2000/OSD-Version, in der KDCMON abläuft. Sehen Sie dazu im Anhang den Abschnitt „Subsystem KDCMON“ auf Seite 240.

Das Programm KDCMON erwartet folgende Steuerparameter:

BUFPAG=size
: gibt die Größe des Puffers in KDCMON in Einheiten von 4 KB an.

  erlaubte Werte: 1 bis 7
  Standardwert: 2 (= empfohlener Wert)

BUFFER=number
: gibt die Anzahl der Puffer in KDCMON an.

  erlaubte Werte: 2 bis 8
  Standardwert: 4 (= empfohlener Wert)

{TIME= mmm | START | BREAK}
bestimmt die Laufzeit von KDCMON.

Die Angabe einer der Alternativen TIME=*mmm*, START oder BREAK beendet die Eingabe der Steuerparameter.

TIME=mmm Laufzeit von KDCMON angegeben in Minuten. Nach der angegebenen Zeit beendet sich KDCMON.

Minimalwert: 1
Maximalwert: 150

START KDCMON läuft 30 Minuten. Entspricht der Angabe TIME=30

Standardwert: START

BREAK KDCMON läuft so lange, bis RESUME eingegeben wird.



ACHTUNG!
KDCMON darf maximal 150 Minuten laufen, sonst kann das Auswertungstool KDCEVAL die Daten nicht auswerten.

Die Puffer haben in KDCMON die gleiche Bedeutung wie in COSMOS.

Das Programm SYSPRG.KDCMON.*nnn* darf während einer Datenerfassung nicht beendet werden, anderenfalls ist das Subsystem KDCMON nicht mehr verfügbar und muss entladen werden.

KDCMON beenden

KDCMON beendet sich nach Ablauf der vorgegebenen Zeit für Parameter TIME=*mmm* und TIME=START, oder nach der Eingabe von RESUME bei Parameter TIME=BREAK. Das Subsystem KDCMON bleibt aber weiterhin geladen und kann mit dem DSSM-Kommando

```
/STOP-SUBSYSTEM KDCMON
```

entladen werden, wenn zu diesem Zeitpunkt keine Datenerfassung läuft. Soll KDCMON entladen werden, wenn die Datenaufzeichnung in einer der UTM-Anwendungen noch nicht abgeschlossen ist, dann muss

```
/STOP-SUBSYSTEM KDCMON,FORCED=YES
```

angegeben werden.

Datenerfassung einschalten und ausschalten

Nach dem erfolgreichen Start von KDCMON können Daten von allen UTM-Anwendungen aufgezeichnet werden, die auf dem jeweiligen Rechner laufen.

KDCMON erfasst auch BCAM-Wartezeiten: BCAM erfasst bei eingeschaltetem KDCMON für Eingabenachrichten die Wartezeit, die sie beim Transportsystem BCAM verbringen, bis *open*UTM sie bei BCAM abholt. Bei Verbindungen mit strengem Dialog werden für alle Eingabenachrichten die Wartezeiten erfasst. Bei anderen Verbindungen wird in Hochlastsituationen eine statistische Auswahl getroffen. *open*UTM übernimmt diese Wartezeit pro Nachricht von BCAM und schreibt sie in die KDCMON-Sätze. KDCEVAL ermittelt daraus den maximalen, minimalen und den mittleren Wert (Angabe in Sekunden). Diese Werte werden in der Liste WAIT (siehe Seite 215) ausgegeben.

Mit Hilfe der UTM-Administration wird gesteuert, von welchen UTM-Anwendungen KDCMON Daten aufgezeichnet.
Die Datenerfassung inklusive Erfassung der BCAM-Wartezeiten können Sie mit dem folgenden Administrations-Kommando ein- und ausschalten:

```
KDCDIAG KDCMON={ ON | OFF }
```

Diese Administrationsfunktion steht auch an der Programmschnittstelle KDCADMI und beim Administrationsprogramm WinAdmin zur Verfügung.

Man kann während eines KDCMON-Laufes für eine Anwendung die Erfassung mehrfach ein- und wieder ausschalten. Dabei sind bis zu 10 Erfassungs-Zeiträume möglich.

Der UTM-Administrator kann jederzeit mit Hilfe des Kommandos

```
KDCINF SYSPARM
```

feststellen, ob Daten erfasst werden oder nicht.

Wenn *open*UTM beim Einschalten feststellt, dass die KDCMON-Funktion nicht verfügbar ist, dann wird folgende Meldung mit dem Standard-Ziel SYSLOG ausgeben:

```
K080 KDCMON-Funktion nicht verfuegbar
```

Mögliche Ursache:
Das Subsystem KDCMON wurde nicht gestartet oder ist nicht unter $TSOS installiert worden.

Wenn *open*UTM bei laufender Erfassung feststellt, dass die KDCMON-Funktion nicht mehr verfügbar ist, dann schaltet *open*UTM die Datenerfassung aus und protokolliert dies ebenfalls mit der Meldung K080.

*open*UTM protokolliert das Ein- und Ausschalten der Erfassung der BCAM-Wartezeit mit der Meldung K146; Standardziel der Meldung ist SYSLOG. Die Meldung K146 gibt *open*UTM auch aus, wenn beim Lesen der BCAM-Wartezeiten ein Fehler auftritt. Als Diagnoseunterlage erzeugt *open*UTM dann einen UTM-Dump mit Dump-Fehlercode ASIS70, UMES02 oder WAIT61. Die Anwendung läuft ohne Erfassung der BCAM-Wartezeit weiter.

13.2.2 Daten auswerten

Zur Auswertung der Daten von KDCMON müssen diese zunächst mit dem Tool KDCPMSM von PAM- bzw. BTAM-Format in SAM-Format umgewandelt werden.
Danach müssen die Sätze nach dem Zeitstempel, der in jedem Satz vorhanden ist, sortiert werden. Dies ist notwendig, da nicht immer garantiert ist, dass die Sätze zeitlich sortiert in der Datei stehen.

13.2.2.1 Daten in SAM-Format umwandeln und sortieren

Zum Umwandeln und Sortieren wird dem Anwender die Prozedur PAMSAM in der Prozedur-Bibliothek SYSPRC.UTM.052 ausgeliefert.

Vor dem Prozeduraufruf müssen Sie zunächst die KDCMON-Datei zuweisen:

- Plattendateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile,ACCESS-METHOD=UPAM
```

- Banddateien

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=kdcmonfile,ACCESS-METHOD=BTAM

/MODIFY-FILE-ATTRIBUTES FILE-NAME=kdcmonfile  -
/     ,SUPPORT=*TAPE(VOLUME=xxxxxx,DEVICE-TYPE=TAPE-C4)
```

Die Prozedur PAMSAM rufen Sie wie folgt auf:

```
CALL-PROCEDURE NAME=SYSPRC.UTM.052(PAMSAM),PROCEDURE-PARAMETERS=(
               [,KDCPMSM=kdcpmsm-progname][,SAMFILE=samfile]
               [,SORT=sortprogram])
```

Bedeutung der Parameter und Standardwerte

kdcmonfile — PAM- bzw. BTAM-Datei mit den aufgezeichneten Daten, die umgesetzt wird.

kdcpmsm-progname — Name des Tools KDCPMSM in Datei SYSPRG.KDCMON.*nnn*.KDCPMSM mit *nnn* = 130, 140 und 150 beim Einsatz in BS2000/OSD V4.0, V5.0 und V6.0.

samfile — Name der Ausgabedatei im SAM-Format, in die PAMSAM die nach dem Zeitstempel sortierten Datensätze schreiben soll.

Standardwert: KDCMON.SAMFILE

sortprogram Name des BS2000-Dienstprogramms SORT.

Standardwert ist $TSOS.SORT, d.h. das BS2000-Dienstprogramm SORT steht unter der Kennung $TSOS zur Verfügung

Während des Ablaufs der Prozedur wird die Datei KDCMON.WORK erzeugt, die nach dem Sortierlauf wieder gelöscht wird.

Nach dem Prozedurlauf stehen die in SAM-Format umgewandelten und nach dem Zeitstempel sortierten Daten in der Datei *samfile*, die im Parameter SAMFILE angegeben wurde.

Bei fehlerhafter Beendigung des Tools KDCPMSM wird der Prozess-Schalter 3 auf 'on' gesetzt.

13.2.2.2 Daten mit dem Tool KDCEVAL auswerten

Mit KDCEVAL können die sortierten Daten ausgewertet werden. Die Auswertung kann im Dialog und im Batch durchgeführt werden. KDCEVAL benötigt zum Ablauf die Eingabe von Steuerparametern.

In einem Auswertungslauf können nur Daten *einer* Anwendung bearbeitet werden. Der Anwender hat die Möglichkeit, durch Angabe eines gewünschten Zeitintervalls (Parameter TIME=) die Auswertung auf einen Teil der erfassten Daten zu beschränken. Wurde während des Auswertungsintervalls für die Anwendung mehrfach die Datenerfassung ein- und ausgeschaltet, so wird für jeden Erfassungszeitraum (von KDCMON=ON bis KDCMON=OFF) eine getrennte Auswertung durchgeführt, wobei maximal 10 solcher Erfassungszeiträume möglich sind.

Das Auswertungstool KDCEVAL wird gestartet mit:

```
/SET-FILE-LINK LINK-NAME=KDCMON,FILE-NAME=samfile                  -
                               ,BUFFER-LENGTH=BY-CATALOG           -
                               ,BLOCK-CONTROL-INFO=BY-CATALOG

/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=SYSLNK.UTM.052.UTIL,ELEM=KDCEVAL)
```

Bedeutung des Parameters

samfile Name der Ausgabedatei von PAMSAM,
d.h. die SAM-Datei mit den sortierten Datensätzen.

Nach dem Start des Auswertungsprogramms im Dialog fordert KDCEVAL mit der folgenden Meldung die Eingabe von Steuerparametern an:

```
PLEASE ENTER COMMANDS OR 'HELP' OR 'END'
```

Steuerparameter von KDCEVAL

Das Programm liest die Parameter von SYSDTA. Die einzelnen Kommandos, mit denen Sie die Auswertung steuern können, haben folgendes Format:

APPLINAME appliname
: Name der Anwendung, für die die Auswertung durchgeführt werden soll. Befinden sich in der Datei Daten mehrerer Anwendungen, so muss für jede Anwendung eine separate Auswertung durchgeführt werden. Eine Auswertung für mehrere Anwendungen ist nicht möglich.

TIME FROM={ t1 | START }, TO={ t2 | STOP }
: Zeitangabe zur Definition des Auswertungsintervalls.

  FROM=t1
  : Startzeitpunkt der Auswertung in Sekunden.
  Die Zeitangabe erfolgt relativ zum Start von KDCMON.

  FROM=START
  : Es wird vom Anfang der Datei an ausgewertet.

  TO=t2 Endzeitpunkt der Auswertung in Sekunden.
  Die Zeitangabe erfolgt relativ zum Start von KDCMON.

  TO=STOP
  : Es wird bis zum Dateiende ausgewertet.

  Für *t1* und *t2* gilt:

  Minimalwert: 0
  Maximalwert: 99999999

LIST { (liste$_1$, liste$_2$,...,liste$_n$ [,TABLE]) | (STD [,TABLE]) | (ALL [,TABLE]) }
: liste$_1$, liste$_2$,...,liste$_n$
  Namen der Einzellisten, die ausgewertet werden sollen. Die Namen, die Sie hier angeben können, finden Sie auf Seite 207. Die Listen TRACE und TRACE2 dürfen nicht zusammen angegeben werden.

  STD Diese Auswertung umfasst die Listen TASKS, SUMM, TIMES und TCLASS.

  ALL Die Auswertung umfasst alle Listen außer TRACE und TRACE2.

  Sollen zusätzlich TRACE oder TRACE2 ausgewertet werden, dann geben Sie LIST (ALL, TRACE) bzw. LIST (ALL, TRACE2) an.

  Werden ALL oder STD ohne TABLE angegeben, dann können die runden Klammern weggelassen werden.

TABLE
Wird zusätzlich TABLE angegeben, dann werden die Listen in einem Tabellenformat erzeugt, das auf einem PC mit Excel oder einem anderen Tabellenkalkulator bearbeitet werden kann, siehe Seite 205. TABLE wirkt nur auf die Einzellisten TASKS, TIMES, TCLASS, TACCL, TACPT und TACLIST.

OPTION **D**ECIMAL-**S**EPARATOR={ **C**OMMA | **P**OINT }

DECIMAL-SEPARATOR=COMMA
Als Dezimaltrennzeichen wird das Komma verwendet.

DECIMAL-SEPARATOR=POINT
Als Dezimaltrennzeichen wird der Punkt verwendet, Standardwert.

END Mit diesem Kommando wird die Parametereingabe beendet.

Bei Auswertungen im Dialog kann auch das Kommando HELP eingegeben werden. Es werden dann die Syntax der Kommandos und die möglichen Listennamen ausgegeben.

Fehler und Meldungen

- Bei fehlerhafter Beendigung von KDCEVAL wird der Prozess-Schalter 3 auf 'on' gesetzt.
- Fehlt eines der Kommandos APPLINAME, TIME oder LIST, so wird die Auswertung mit folgender Fehlermeldung abgebrochen:

  ```
  MANDATORY COMMAND MISSING
  ```

- Bei einem Syntaxfehler erscheint die folgende Meldung und das fehlerhafte Kommando wird angezeigt:

  ```
  ERROR IN COMMAND
  ```

- Sind die Zeitangaben *t1* und *t2* inkonsistent, so wird folgende Meldung ausgegeben:

  ```
  KDCEVAL: WRONG TIME INPUT
  ```

- Werden in der Datei keine Sätze für die Anwendung gefunden oder sind für das Auswertungsintervall keine Daten vorhanden, so wird eine der folgenden Meldungen ausgegeben:

  ```
  NO EVALUATION : NO RECORD WITH APPLINAME FOUND
  ```

  oder

  ```
  NO EVALUATION : NO RECORD IN TIME_INTERVAL
  ```

- Bei einem DMS-Fehler werden folgende Meldungen ausgegeben:

  ```
  KDCEVAL: DMS-ERROR Dxxx FOR INPUT FILE
  KDCEVAL: NO EVALUATION
  ```

  (Dxxx = DMS-Errorcode)

- Versionsprüfung:

  Die Auswertung der KDCMON-Daten durch KDCEVAL ist nur möglich, wenn KDCEVAL die gleiche UTM-Version wie der UTM-Systemcode hat. KDCEVAL prüft die Version der KDCMON-Daten. Erkennt KDCEVAL eine unzulässige Version, bricht KDCEVAL die Auswertung mit der folgenden Meldung ab:

  ```
  NO EVALUATION: INPUT FILE FROM INVALID UTM VERSION
  ```

Ergebnis der Auswertung mit KDCEVAL

Das Ergebnis der Auswertung schreibt KDCEVAL in die Dateien einer Dateigenerationsgruppe (FGG) mit dem Namen:

KDCMON.*appliname*

Die FGG enthält maximal 10 Generationen. Beim Start der Auswertung wird eine bereits vorhandene Dateigeneration dieses Namens gelöscht.

Wenn während eines KDCMON-Laufs die Datenerfassung mehrmals ein- und ausgeschaltet wurde, schreibt KDCEVAL mit jedem Einschalten in eine eigene Datei.

13.2.3 Auswertungsdaten auf dem PC bearbeiten

Wenn Sie bei KDCEVAL im Steuerparameter LIST zusätzlich zu den Listennamen den Operanden TABLE angeben, dann werden diese Listen in Tabellenform erzeugt. Diese Art der Aufbereitung ist nur für Listen TASKS, TIMES, TCLASS, TACCL, TACPT, und TACLIST möglich.

Diese so erzeugten Listen können Sie auf dem PC mit einem Tabellenkalkulator wie z.B. Excel bearbeiten und grafisch aufbereiten. Für Excel werden dazu die beiden Makros `kdceval.ftp.xls` und `kdceval.openft.xls` ausgeliefert.

Dazu gehen Sie wie folgt vor:

1. Übertragen Sie die von KDCEVAL erzeugte Listen-Datei mit ftp oder *open*FT auf einen PC. Zusätzlich müssen Sie das passende Makro übertragen: `kdceval.ftp.xls` für die Übertragung mit ftp oder `kdceval.openft.xls` für die Übertragung mit *open*FT.

   Das Makro verlangt, dass die auszuwertende Datei die Endung `.txt` besitzt!

2. Rufen Sie das Makro auf (`kdceval.ftp.xls` bzw. `kdceval.openft.xls)` und lesen Sie die Listen-Datei in Excel ein. Excel erzeugt dann für jede Liste ein eigenes Tabellenblatt sowie ein Zusatzblatt mit Übersichtsinformationen. (Summary-Blatt).

3. Bearbeiten Sie die einzelnen Listen nach Ihren Wünschen, indem Sie z.B. eine Liste sortieren und anschließend in ein Kurven- oder Balkendiagramm umwandeln.

13.2.4 Auswertungslisten

Jede Auswertungsliste besteht aus

- einer Überschrift, die den Namen der Auswertungsliste enthält
- einem Kopf, der für alle Listen identisch ist
- der spezifischen Auswertungsliste

Der Listenkopf hat folgenden Aufbau:

```
NAME OF APPLICATION :appli     DATE                  :yyyy-mm-dd BS2000 VERSION  :Vnn.n  UTM VERSION :V05.2A
COMMENCEMENT TIME   :sss SEC.  KDCMON START          :  hh:mm:ss KDCEVAL VERSION :V05.2A
END TIME            :sss SEC.  APPL.RECORDING START :  hh:mm:ss
SYSTEM INFORMATION  :PROCESSOR HSIBASE Number CPUs  :4 Bitmode :32
```

Die Felder haben folgende Bedeutung (soweit nicht selbsterklärend):

NAME OF APPLICATION
: Name der Anwendung

DATE
: Datum der Datenerfassung mit KDCMON

BS2000 VERSION
: BS2000-Version der Anlage, auf der KDCMON gelaufen ist

COMMENCEMENT TIME
: Anfangszeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt von KDCMON)

KDCMON START
: Startzeit von KDCMON

END TIME
: Endzeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt von KDCMON)

APPL. RECORDING START
: Startzeit der Datenerfassung für die UTM-Anwendung

Bei der TRACE- und TRACE2-Liste enthält END TIME den Wert 999999, wenn man die ganze Datei auswertet (Parameter TIME FROM=START,TO=STOP).

Bei den Bearbeitungszeiten handelt es sich stets um die ELAPSED TIME (Realzeit).

Es sind folgende Einzel-Auswertungen und Kombinationen dieser Auswertungen möglich:

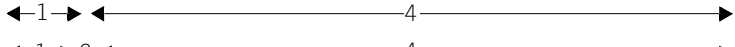
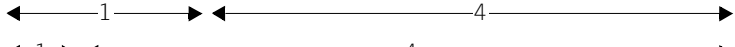
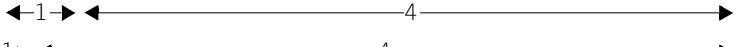
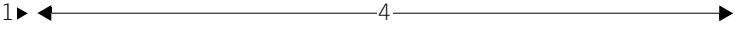
TASKS	UTILIZATION OF THE UTM TASKS
SUMM	TRANSACTION EVALUATION
TIMES	DISTRIBUTION OF PROCESSING TIMES
KCOP	KDCS CALLS STATISTIC
WAIT	WAITING TIMES
TCLASS	EVALUATION OF THE TAC CLASSES
TACCL	TAC SPECIFIC TAC CLASS EVALUATION
TACPT	TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES
TACLIST	TAC SPECIFIC STATISTICS
TRACE	TASK SPECIFIC TRACES
TRACE2	TASK PERFORMANCE TRACES

Im Folgenden werden die einzelnen Auswertungslisten beschrieben.

TASKS: UTILIZATION OF THE UTM TASKS

In dieser Liste wird ein Überblick über die Auslastung der Prozesse der Anwendung gegeben. Außerdem werden der CPU-Verbrauch und die Anzahl der Ein- und Ausgabeoperationen (I/O's) für jeden einzelnen UTM-Prozess aufsummiert und für alle Prozesse der Anwendung angezeigt.

```
1   Program
2   System code
3   Database
4   Bourse Wait
```

TSN	START TIME	TASK UTILIZATION
OPTB	15:10:06.937937	
OPTA	15:10:17.032158	
OPS9	15:10:47.076819	
OPS8	15:11:07.113254	
OPS6	15:13:07.136012	

TSN	CPU-time	Number I/O	Program	System	Database	Bourse Wait
OPTB	752	33	31461	1308	0	279508
OPTA	1713	124	24909	3563	0	269610
OPS9	439	16	101358	1705	0	238972
OPS8	110	4	20770	480	0	219725
OPS6	296	4	2068	624	0	99668
Summ	3310	181				

In der Liste bedeuten:

TSN	TASK SEQUENCE NUMBER des UTM-Prozesses.
START TIME	Zeit des ersten Satzes dieses Prozesses (absolut).
program	Zeitanteil des Anwendungsprogramms im UTM-Prozess. In dieser Zeit ist auch die SVC-Behandlung der SVCs enthalten, die aus den Anwendungsprogrammen aufgerufen werden.
system code	Zeitanteil des UTM-Systemcodes.
data base	Zeitanteil, der zur Bearbeitung der Datenbankaufrufe benötigt wurde. Erfordert die Bearbeitung eines Datenbankaufrufs einen Prozesswechsel, so beinhaltet diese Zeit auch die Wartezeit des UTM-Prozesses.
bourse wait	Zeitanteil, den der Prozess auf neue Aufträge an der Auftragswarteschlange gewartet hat.

Für die Auswertung TASKS ist ein Herabsetzen der Prozess-Anzahl während des Auswertungsintervalls zu vermeiden, denn es führt zu falschen Ergebnissen. In diesem Fall sollten Sie weitere Auswertungsintervalle verwenden.

SUMM: TRANSACTION EVALUATION

In dieser Liste wird ein Überblick über die Vorgänge und Transaktionen für den Auswertungszeitraum gegeben. Es werden nur solche Transaktionen berücksichtigt, die vollständig im Auswertungszeitraum liegen. Zusätzlich gibt das Auswertungstool KDCEVAL den CPU-Verbrauch von allen Teilprogrammläufen aus, die im Auswertungsintervall beendet wurden. Die Liste hat folgendes Format:

```
COUNT OF TRANSACTIONS                          :   240 1)
COUNT OF SERVICES                              :    80 2)
COUNT OF DIALOG STEPS                          :   290
NUMBER OF DIALOG STEPS PER SECOND              :     4
NUMBER OF DB CALLS PER TRANSACTION             :     2
TOTAL CPU-TIME USED IN MSEC                    :  9150 3)
```

1) Die Transaktion KDCDIAG zum Ein- und Ausschalten des Messmonitors wird nicht gezählt.

2) SERVICE = Vorgang

3) In dieser Zeile wird die Summe des CPU-Verbrauchs der einzelnen Teilprogrammläufe ausgegeben. Darin ist auch der Verbrauch im UTM- und Betriebssystemcode enthalten, soweit er innerhalb der Teilprogrammläufe anfällt, sowie die Start- und Endeverarbeitung in *open*UTM. Andere Aktionen der UTM-Prozesse, die nicht direkt zu Teilprogrammläufen gehören, sind nicht enthalten.

TIMES: DISTRIBUTION OF PROCESSING TIMES

In dieser Liste wird in tabellarischer Form eine grobe Verteilung der Bearbeitungszeiten für die Teilprogramme ausgegeben. In diesen Zeiten ist die Wartezeit vor der Bearbeitung durch *open*UTM nicht enthalten.

Die Liste hat folgendes Format:

PROCESSING TIMES (MSEC)	NUMBER	PERCENT
0 - 100	76	58.91
101 - 200	45	34.88
201 - 500	7	5.42
501 - 1000	1	0.77
1001 - 2000	0	0.00
2001 - 5000	0	0.00
5001 - 10000	0	0.00
10001 - 20000	0	0.00
20001 - 50000	0	0.00
50001 - 100000	0	0.00
> 100000	0	0.00

In dieser Liste ist die Anzahl der vollständigen Teilprogrammläufe und der prozentuale Anteil für die jeweilige Zeitklasse angegeben.

KCOP: UTM CALLS STATISTIC

In dieser Tabelle ist für die UTM-Aufrufe angegeben, wie oft sie im Auswertungszeitraum aufgetreten sind.

Unter *others* sind Aufrufe aufgeführt, die nicht in der Liste der von *open*UTM unterstützten Aufrufe enthalten sind.

In dieser Liste sind Aufrufe enthalten, die *open*UTM für interne Bearbeitungen aufruft und die dem Anwender nicht zur Verfügung stehen:

CONT	Aufruf nach der Formatierung oder einer internen Datenbankkommunikation
ADMI	UTM-Administrationsaktion
WAIT	Ende der Bearbeitung eines Programmlaufes
NOOP	Messdaten für DB-Aufrufe schreiben

Die KCOP-Liste hat folgendes Format:

OP	OM	NUMBER
INIT		7662
INIT	PU	0
INIT	MD	0
CONT		1857
CTRL	AB	0
CTRL	PE	0
CTRL	PR	0
CTRL	EC	0
APRO	AM	0
APRO	DM	0
APRO	IN	0
MCOM	BC	0
MCOM	EC	0
MGET		7331
MPUT	NT	8
MPUT	NE	5753
MPUT	PM	0
MPUT	RM	4

OP	OM	NUMBER
SREL	GB	0
GTDA		0
PTDA		0
UNLK	DA	0
UNLK	GB	0
UNLK	US	0
LPUT		0
INFO	LO	0
INFO	CD	0
INFO	CK	0
INFO	DT	0
INFO	SI	0
INFO	FH	0
INFO	GN	0
INFO	PC	0
DADM	RQ	0
DADM	UI	0
DADM	CS	0

OP	OM	NUMBER
MPUT	EM	0
MPUT	HM	0
MPUT	ES	0
MPUT	GC	0
FGET		0
FPUT	NT	0
FPUT	NE	0
FPUT	RP	0
DPUT	NI	0
DPUT	NT	0
DPUT	NE	0
DPUT	+I	0
DPUT	-I	0
DPUT	+T	0
DPUT	-T	0
DPUT	QT	0
DPUT	QE	0
DPUT	QI	0
DPUT	RP	0
DGET	FT	0
DGET	NT	0
DGET	BF/BN	0
DGET	PF/PN	0
QCRE	NN	0
QCRE	WN	0
QREL	RL	0
SPUT	DL/MS/ES	42284
SPUT	GB	0
SPUT	US	0
SGET	KP	2917
SGET	GB	0
SGET	RL	0
SGET	US	0

OP	OM	NUMBER
DADM	DL	0
DADM	DA	0
PADM	OK	0
PADM	PR	0
PADM	AT	0
PADM	AC	0
PADM	CA	0
PADM	CS	0
PADM	AI	0
PADM	PI	0
SIGN	ST	0
SIGN	ON	0
SIGN	CP	0
SIGN	CK	0
SIGN	OF	0
SIGN	OB	0
SIGN	CL	0
SMSG		0
NOOP		1111
WAIT		5762
ADMI		80
RSET		132
PGWT	CM	0
PGWT	RB	0
PGWT	KP	0
PGWT	PR	0
PEND	SP	1900
PEND	RE	65
PEND	FI	323
PEND	ER	0
PEND	FR	0
PEND	KP	5370
PEND	RS	4

OP	OM	NUMBER
SREL	LB	0
OTHERS		0

OP	OM	NUMBER
PEND	PS	0
PEND	FC	0
PEND	PA/PR	0

WAIT: WAITING TIMES

Zur Ermittlung von Stausituationen fügt *open*UTM bei eingeschaltetem KDCMON in regelmäßigen Zeitabständen Pseudoaufträge in die Auftragswarteschlange ein. Mit Hilfe des Zeitpunktes, zu dem der Auftrag eingefügt wurde (Zeitstempel absolut) und dem Bearbeitungszeitpunkt kann die Wartezeit der Aufträge in der UTM-Warteschlange ermittelt werden. Der Zeitabstand zwischen den einzelnen Pseudoaufträgen beträgt etwa 10 Sekunden. Er kann aber auch größer sein, da die Initiierung dieser Pseudoaufträge an bestimmte UTM-Ereignisse gekoppelt ist.

In der WAIT-Liste wird Folgendes protokolliert:

- In der Spalte WAITING TIME wird die ermittelte Wartezeit für jeden Pseudoauftrag in Sekunden ausgegeben.
- Für diese Wartezeiten berechnet das Auswertungstool KDCEVAL zusätzlich den Maximal-, Minimal- und Mittelwert in Sekunden und zeigt diese Werte unter UTM WAITING TIMES an.
- In der Spalte NUMBER OF TASKS wird die Anzahl der Prozesse, die zu diesem Zeitpunkt in der Anwendung zur Verfügung standen, ausgegeben.
- Die BCAM-Wartezeiten werden in der Spalte BCAM WAITING TIMES in Sekunden ausgegeben. BCAM misst die Zeit, die Eingabenachrichten an *open*UTM im Transportsystem BCAM verbringen, bevor *open*UTM sie abholt.

  Bei NUMBER OF MESSAGES wird die Anzahl der KDCMON-Sätze im Messintervall angezeigt, die jeweils zu einem Teilprogrammlauf erzeugt wurden, und für deren Nachricht BCAM eine Wartezeit geliefert hat. Beim Einschalten der KDCMON-Aufzeichnung (und damit auch der Zeiterfassung bei BCAM) sind i.a. bereits Nachrichten im Nachrichtenpool von BCAM. Wenn *open*UTM diese Nachrichten bei BCAM abholt, erhält *open*UTM für sie keine Wartezeit. Diese Nachrichten werden im Wert NUMBER OF MESSAGES nicht mitgezählt.

Wenn die Wartezeit zu lang wird, sollte die Anzahl der UTM-Prozesse erhöht werden.

Die WAIT-Liste hat folgendes Format:

TIME STAMP	WAITING TIME	NUMBER OF TASKS
10:33:35.742	0.018	1
10:33:45.781	0.008	2
10:33:55.841	0.007	3
10:33:66.027	0.008	3
10:33:76:087	0.008	3
10:33:86:112	0.007	3
10:33:96:123	0.007	3

```
                        UTM WAITING TIMES:

TIME STAMP :  10:33:35.742    WAITING TIME MAXIMUM :        0.018
TIME STAMP :  10:33:41.740    WAITING TIME MINIMUM :        0.007
NUMBER OF ENTRIES:       7    WAITING TIME AVERAGE :        0.009

                        BCAM WAITING TIMES:

TIME STAMP :  10:23:35.742    WAITING TIME MAXIMUM :        0.728
TIME STAMP :  10:39:35.740    WAITING TIME MINIMUM :        0.027
NUMBER OF MESSAGES:    200    WAITING TIME AVERAGE :        0.125
```

TCLASS: EVALUATION OF THE TAC CLASSES

Die TCLASS-Liste enthält in tabellarischer Form einen Überblick über die Auftragsbearbeitung von TACs der einzelnen TAC-Klassen (1 bis 16). In der Auswertung sind in der TAC-Klasse 0 alle die Dialog-TACs zusammengefasst, denen bei der Generierung mit KDCDEF keine TAC-Klasse zugeordnet wurde.

Bei der UTM-Generierung kann der Anwender festlegen, wieviele Prozesse zu einem Zeitpunkt maximal für eine TAC-Klasse arbeiten dürfen. Ist diese Anzahl erreicht, so werden weitere Aufträge in eine TAC-Klassen-spezifische Warteschlange eingereiht.

TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT			AVERAGE WAIT TIME (IN MSEC)	MAXIMUM WAIT TIME (IN MSEC)	MINIMUM WAIT TIME (IN MSEC)
		NUMBER CALLS	WAIT TIME=0	WAIT TIME>0			
0	21	30.87					
1	2	2.94	100.00	0.00	0	0	0
2	4	5.88	75.00	25.00	3000	5000	1000
3	3	4.41	100.00	0.00	0	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
8	5	7.35	80.00	20.00	5000	8000	2000
9	2	2.94	90.00	10.00	4000		
10	3	4.41	100.00	0.00	0	0	0
11	4	5.88	75.00	25.00	3000	5000	1000
12	3	4.41	100.00	0.00	0	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
16	5	7.35	80.00	20.00	5000	8000	2000
			90.00	10.00	4000		

41 DIALOG TACS WERE CALLED
27 ASYNCHRONOUS TACS WERE CALLED

Die TCLASS-Liste enthält folgende Angaben:

- In der Spalte TAC-CLASS wird für eine TAC-Klasse die Anzahl der TAC-Aufrufe im Auswertungszeitraum angegeben.

- Die Spalte DISTRIBUTION IN PERCENT enthält Prozentwerte.
  Die Unterspalte NUMBER CALLS gibt den prozentualen Anteil der Aufrufe einer TAC-Klasse zur Anzahl aller Dialog-TAC-Aufrufe an. Die nächsten beiden Spalten enthalten eine prozentuale Aufteilung der Aufrufe dieser TAC-Klasse in
  - Aufrufe, die sofort bearbeitet wurden (WAITTIME=0) und
  - Aufrufe, die in eine TAC-klassenspezifische Warteschlange eingereiht werden mussten (WAITTIME>0)
- In der Spalte AVERAGE WAIT TIME wird die mittlere Wartezeit pro TAC-Klasse angezeigt.
- Die Werte in den Spalten AVERAGE / MINIMUM / MAXIMUM WAIT TIME beziehen sich auf die Aufträge, die *open*UTM vorübergehend in eine TAC-klassenspezifische Warteschlange verdrängt hat. Es wird die mittlere, minimale bzw. maximale Wartezeit eines Auftrags pro TAC-Klasse angezeigt.



Die mittlere Wartezeit von Aufträgen pro TAC-Klasse kann im laufenden Betrieb einer Anwendung auch mit dem Administrationskommando KDCINF TACCLASS oder mit der entsprechenden Funktion bei WinAdmin oder KDCADMI abgefragt werden.

Wartezeit bei Dialogaufträgen

Bei Dialogaufträgen ist die Wartezeit die Zeitspanne zwischen der Entgegennahme des Auftrags durch die Anwendung (Abholen des Auftrags von der Warteschlange der Anwendung) und dem Start des Teilprogramms. Es kann auch zwischen einzelnen Teilprogrammen zur Verdrängung kommen.

Wartezeit bei Asynchronaufträgen

*open*UTM erfasst auch die Wartezeit von Asynchronaufträgen, die sich wie folgt definiert:

Asynchronauftrag	**Definition „Wartezeit“**
Eingabe Asynchron-TAC	Zeitspanne zwischen der Entgegennahme des Auftrags durch *open*UTM und dem Starten des Asynchronvorgangs
FPUT-Aufruf im Teilprogramm	Zeitspanne zwischen dem Ende der Transaktion, in der der FPUT-Auftrag ausgeführt wurde, und dem Starten des Asynchronvorgangs
DPUT-Aufruf im Teilprogramm	Zeitspanne zwischen der Umwandlung des DPUT in einen FPUT und dem Starten des Asynchronvorgangs

Wurde der Asynchronauftrag nicht im aktuellen Anwendungslauf erzeugt, so wird als asynchrone Wartezeit immer die Zeitdifferenz zwischen dem Start der Anwendung und dem Start des Asynchronauftrags genommen.

TACCL: TAC SPECIFIC TAC CLASS EVALUATION

In der Liste TACCL sind die gleichen Informationen wie in der TCLASS-Liste aufgeführt, nur aufgegliedert auf die einzelnen Transaktionscodes. Es sind alle TACs aufgeführt, die im Auswertungszeitraum aufgerufen wurden. Die TACs sind in der Reihenfolge des ersten Auftretens aufgeführt. Die Bedeutung der einzelnen Spalten kann der Beschreibung des TCLASS-Listenformates entnommen werden.

TAC	TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT			AVERAGE WAIT TIME (IN MSEC)
			NUMBER CALLS	WAIT TIME=0	WAIT TIME> 0	
TAC1	0	10	22.50			
TAC2	5	5	11.25	60.00	40.00	1000
TAC3	2	1	2.50	0.00	100.00	3256
TAC4	13	12	23.40	40.00	60.00	2000
TAC5	15	17	26.20	55.00	45.00	4000
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Für TACs der TAC-Klasse 0 werden keine Angaben für WAIT TIME eingetragen.

TACPT: TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES

In dieser Tabelle sind für alle im Auswertungszeitraum bearbeiteten TACs die minimale (MIN), die maximale (MAX) und die mittlere (MEAN) Bearbeitungszeit aufgeführt. Es werden nur solche TACs aufgenommen, deren Start- und Beendigungszeitpunkt innerhalb des Auswertungszeitraums liegen. Die Liste hat folgendes Format:

TAC	PROCESSING TIME PER TAC (IN MSEC)		
	MEAN	MIN	MAX
TAC1	150000	125000	175000
TAC2	35000	19000	45000
TAC3	24500	20000	29000

Die Tabelle ist nach mittleren Bearbeitungszeiten fallend sortiert. Angezeigt werden nur TACs mit einer mittleren Bearbeitungszeit > 0.

TACLIST: TAC SPECIFIC STATISTICS

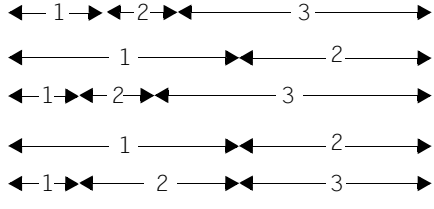
In dieser Liste werden die folgenden TAC-spezifischen Informationen erfasst:

- die durchschnittliche KB-Größe (Spalte AVERAGE SIZE OF KB)
- die durchschnittliche Anzahl der Datenbankaufrufe, getrennt nach
  - Aufrufen aus den Anwendungsteilprogrammen (Spalte FORM USER) und
  - Aufrufen, die zur Kommunikation zwischen *open*UTM und dem Datenbanksystem dienen (Spalte FROM SYSTEM)
- die Aufteilung der Bearbeitungszeit in

  1: program
  2: system code
  3: data base (die Bedeutung dieser Zeiten ist die gleiche wie in der TASKS-Liste)

Die Liste hat folgendes Format:

TAC	NUMBER CALLS	AVERAGE DB CALLS		AVERAGE SIZE OF KB
		FROM USER	FROM SYSTEM	
TAC1	5	15	2	256
TAC2	18	0	0	1024
TAC3	3	200	2	3000
TAC4	70	0	0	1024
TAC5	500	32	2	30000
.	.	.	.	.
.	.	.	.	.



Die Liste ist nicht sortiert, die TACs sind in der Reihenfolge aufgelistet, in der sie in der Datei zuerst auftreten.

Es werden nur die TACs berücksichtigt, die einen vollständigen Börsenzyklus hatten.

TRACE: TASK SPECIFIC TRACES

Zur genaueren Analyse des Ablaufs einer UTM-Anwendung können TRACE-Listen erstellt werden. In diesen Listen werden in zeitlicher Reihenfolge für die einzelnen UTM-Prozesse alle UTM-Aufrufe aufgelistet.

In einer Liste können aus Platzgründen die Daten von maximal 6 Prozessen enthalten sein. Wenn für den Auswertungszeitraum Daten für mehr als 6 Prozesse existieren, so erzeugt das Auswertungstool KDCEVAL eine zweite TRACE-Liste mit den nächsten maximal 6 Prozessen und hängt sie in der gleichen Datei an die erste Liste an. Dadurch ist eine Analyse des zeitlichen Ablaufs von maximal 12 Prozessen möglich.

Sind in dem Auswertungszeitraum Daten von mehr als 12 Prozessen (jedoch höchstens 25) vorhanden, so erstellt KDCEVAL die Auswertungslisten für 12 Prozesse. Für die restlichen Prozesse wertet KDCEVAL keine Daten aus und protokolliert dies durch eine Meldung mit Angabe der TSNs.

Die Liste ist in zeitlicher Reihenfolge sortiert.

Die Spalte TIME STAMP enthält den Zeitstempel des entsprechenden Aufrufs, der protokolliert wurde (in Mikrosekunden).

Die Liste TRACE erfasst folgende Ereignisse und Daten:

- den aufgerufenen Transaktionscode (TAC)
- die Transaktions-ID. In *open*UTM wird für jede Transaktion eine eindeutige Transaktions-ID vergeben. Dieses Kennzeichen wird auch den angeschlossenen Datenbanken an der UTM-DB-Schnittstelle übergeben. Damit wird es u.a. möglich, Traces der Datenbank mit diesen Traces von *open*UTM zu koppeln und Zusammenhänge zwischen UTM- und DB-Abläufen herzustellen. Die Transaktions-ID besteht aus vier Teilen:

  SC Session Counter: er nummeriert die Anwendungsläufe. Nach Neugenerierung ist er 1, bei jedem Start der Anwendung wird er um 1 hochgezählt.

  VC Vorgangs Counter: er nummeriert die Vorgänge innerhalb eines Anwendungslaufes und läuft bis 16 777 216 (2^{24}).

  TC Transaction Counter: er nummeriert die Transaktionen innerhalb eines Vorgangs und läuft bis 32 768 (2^{15}).

  VN Vorgangs Nummer: das ist bei einem Dialogvorgang die bei der UTM-Generierung vergebene Nummer des USERs, bei einem Asynchronvorgang die beim Anwendungsstart vergebene Nummer des UTM-Prozesses.

  Diese vier Teile werden nach dem KDCS-Aufruf INIT protokolliert.
  Für den Anwender sind die Angaben zu VC und TC interessant.

- Alle UTM-Aufrufe mit ihren Operationsmodifikationen. Auch UTM-interne Aufrufe (WAIT, CONT, ...) werden aufgeführt. Sehen Sie auch die KCOP-Liste.

  Zusätzlich wird protokolliert:

  – KCMF bei Aufrufen, bei denen KCMF relevant ist

  – KCRN bei Aufrufen, bei denen KCRN relevant ist

  – KCLT bei den Aufrufen PADM/DADM

  – bei einem Abbruch mit PEND ER/ FR als Diagnoseinformation:
    – der TAC des Teilprogramms, das den Abbruch verursacht hat
    – die Returncodes KCRCDC und KCRRCC
    – VC und TC, für die Zuordnung zum abgebrochenen Vorgang

  – bei einem Aufruf PEND RS als Diagnoseinformation:
    – der TAC des aktuellen Teilprogramms
    – VC und TC, für die Zuordnung zum abgebrochenen Vorgang

  – bei DB-Aufrufen FITA und CATA
    – VC und TC, für die Zuordnung zum abgebrochenen Vorgang

- Alle Datenbankaufrufe, wenn die UTM-Anwendung koordiniert mit einem Datenbanksystem zusammenarbeitet (generiert mit KDCDEF-Steueranweisung DATABASE).

  Aufrufe an ein Datenbanksystem ≠ SESAM werden mit folgendem Eintrag protokolliert:

  opcode db-time

  Dabei ist *opcode* der Operationscode des Datenbankaufrufs gemäß IUTMDB-Schnittstelle. Sehen Sie dazu die Beschreibung der DB-DIAGAREA im *open*UTM-Handbuch „Meldungen, Test und Diagnose (BS2000/OSD)“, Kapitel UTM-Dump. *db-time* gibt an, wieviel Zeit zur Bearbeitung des Datenbankaufrufs benötigt wurde (Realzeit in Millisekunden)

  Beispiel: `FITA  115`

  Wenn auf der Seite des DBH der SESCOS eingeschaltet ist, werden Aufrufe an das Datenbanksystem SESAM ab SESAM/SQL V1.1 durch folgenden Eintrag protokolliert, der die Zuordnung zwischen TRACE-Liste und SESCOS-Auswertung ermöglicht:

  ppxyttttnnnmmmmmmmmm

  pp Operationscode des DB-Aufrufs als Hexawert, z.B. 14=DBFITA
  Werte siehe DB-DIAGAREA

  x Konfigurationsname des SESAM-DBH
  Wertebereich: ␣, A-Z, 0-9
  Wert ‘-‘: Information wurde von SESAM nicht versorgt

y Kommunikationsname des SESAM-DBH
Wertebereich: ␣, A-Z, 0-9

tttt TSN der DBH-Task

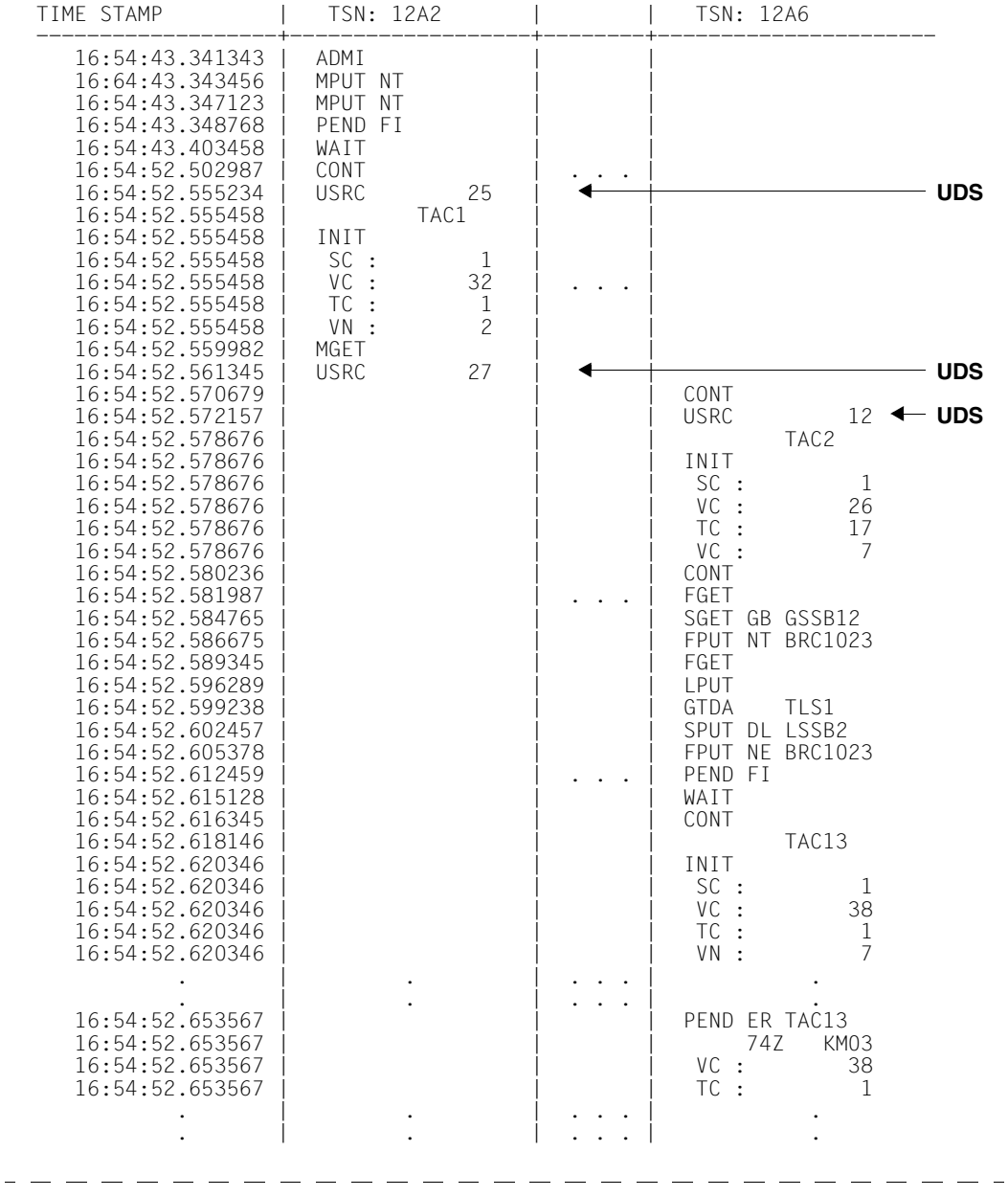
nnn laufende Nummer der SESCOS-Datei

mmmmmmmm
laufende Nummer in der SESCOS-Datei.
Die Zähler *nnn* und *mmmmmmmm* werden mit führenden Leerzeichen angegeben.

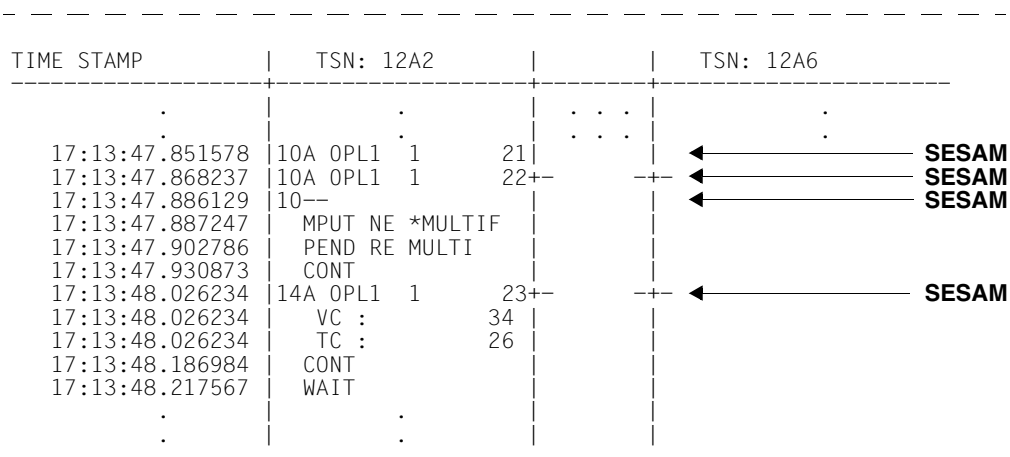
Solange kein Prozesswechsel stattfindet, sind alle Aufrufe zur Bearbeitung eines Dialogschrittes in der gleichen TSN-Spalte hintereinander aufgeführt. Nach PEND PA/PR kann ein Prozesswechsel nur bei einem Wechsel der TAC-Klasse auftreten. Die Unterbrechung eines Prozesses durch das Betriebssystem wird dadurch sichtbar, dass mitten in der Bearbeitung eines Dialogschrittes die Aufrufe in einer anderen Prozess-Spalte fortgesetzt werden.

Es folgt ein Beispiel für die Darstellung der TRACE-Liste. Sie zeigt einen Ausschnitt aus dem Ablauf einer UTM-Anwendung mit Datenbankaufrufen an UDS und SESAM.

Die Auswertungsliste TRACE hat folgendes Format:



Fortsetzung:



Im Beispiel wurden zum besseren Verständnis die Datenbankaufrufe durch Pfeile gekennzeichnet, die in der Original TRACE-Liste nicht enthalten sind.

TRACE2: TASK PERFORMANCE TRACE

Die Auswertungsliste TRACE2 listet die wichtigsten Ereignisse in den Teilprogrammen der Anwendung sequenziell auf. Da die Auswertung nicht wie in der Liste TRACE in Spalten für die UTM-Prozesse erfolgt, kann die Liste TRACE2 beliebig viele Prozesse auswerten. TRACE2 enthält zusätzlich zu den Einträgen der Auswertung TRACE wichtige Daten zur Performanceanalyse.

Die Einträge in der Auswertung sind in zeitlicher Reihenfolge sortiert. Die Spalte TIME STAMP enthält den Zeitstempel des Ereignisses (Genauigkeit: Mikrosekunden).

Die Auswertungsliste TRACE2 erfasst folgende Ereignisse und Daten:

- Start des Teilprogramms als Eintrag `strt >>>` *tac* mit
  - Transaktionscode des Teilprogramms
  - TAC-Klasse
  - aktueller I/O- und CPU-Stempel
  - Wartezeit der Nachricht in BCAM
  - Wartezeit des Auftrags in der TAC-Klasse
- Alle UTM-Funktionsaufrufe mit Operationscode und -modifikation und zusätzlich folgende Informationen:
  - KCMF bei Aufrufen, bei denen KCMF relevant ist
  - KCRN bei Aufrufen, bei denen KCRN relevant ist
  - KCLT bei den Aufrufen PADM und DADM
  - Bei PEND-Aufrufen mit KCOM = ER/FR/RS und bei den Datenbankaufrufen FITA und CATA die Transaction-ID (SC,VC,TC und VN) für die Zuordnung zum abgebrochenen Vorgang.
  - Wenn KCRCCC ≠ 0, die Returncodes KCRCCC und KCRCDC und die Transaction-ID (SC,VC,TC und VN).
- Alle Datenbankaufrufe, wenn die UTM-Anwendung koordiniert mit dem DB-System zusammenarbeitet. Die Aufrufe an das Datenbanksystem werden protokolliert mit einem Eintrag:

  DBCL *opcode db-time*

  opcode und *db-time* haben die gleiche Bedeutung wie bei der Liste TRACE.

  Wenn auf der Seite des DBH der SESCOS eingeschaltet ist, wird bei Aufrufen an das Datenbanksystem SESAM zusätzlich eine Information `SESCOS-INFO -->`*xytttnnnmmmmmmmmm* protokolliert. Die Werte ermöglichen die Zuordnung von TRACE2-Listen und SESCOS-Auswertung. Sehen Sie dazu auch die Beschreibung der Felder für die TRACE-Liste auf Seite 223.

- Ende des Teilprogramms als Eintrag `WAIT end<<<<` mit
  - CPU-Verbrauch im Teilprogramm in Millisekunden in Spalte „CPU“
  - I/O-Verbrauch im Teilprogramm in Spalte „I/0“
- Eine logische Maschinennummer, die KDCMON liefert.

  Jede Nummer identifiziert dabei jeweils eine CPU. Die Nummer wird in der Auswertung in der Spalte „ID“ angezeigt. Auf Multiprozessoranlagen kann damit festgestellt werden, welche UTM-Prozesse parallel laufen.

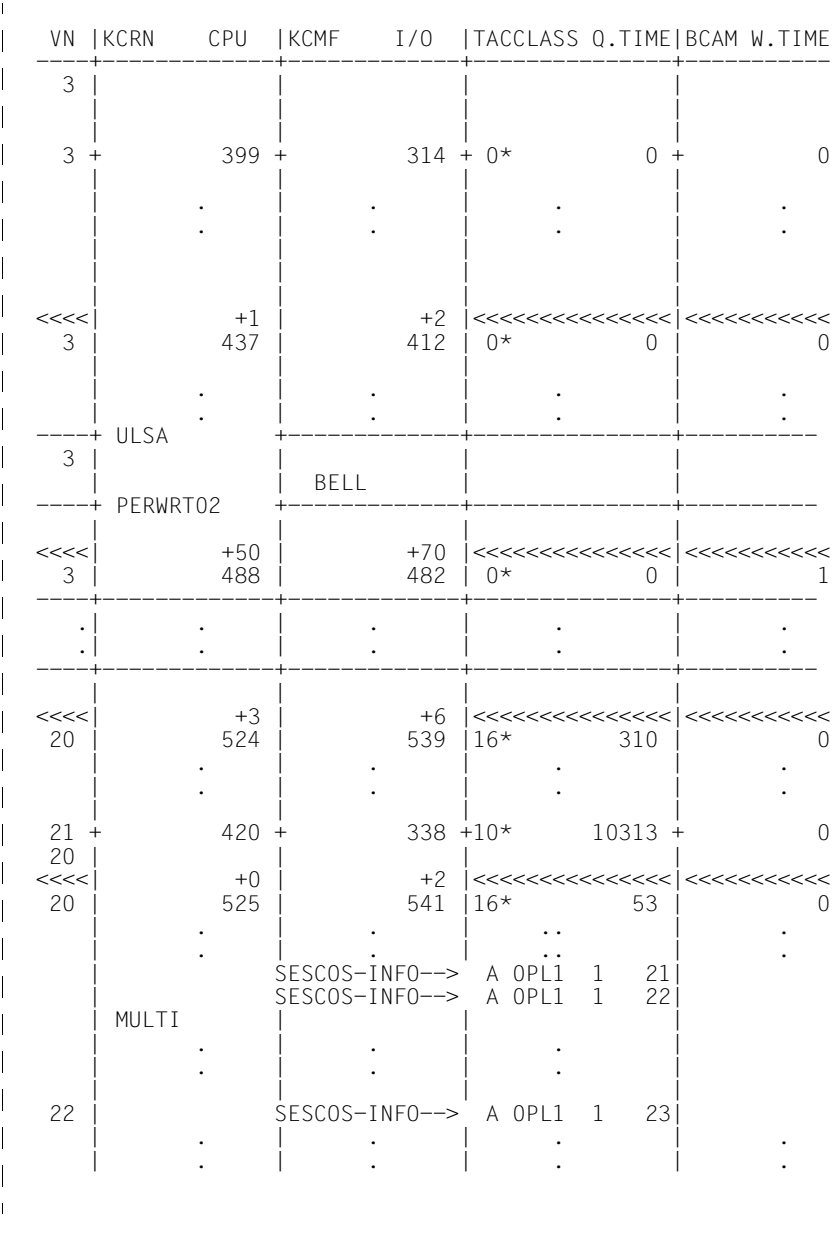
Die Strukturelemente <<<<<< in der Liste erleichtern die Lesbarkeit der Einträge.

Die Auswertungsliste TRACE2 hat das folgende Format. Diese Seite zeigt dabei den linken Teil und die nächste Seite den rechten Teil der Liste:

linker Teil der Liste

```
TIME STAMP        |  TRACE:          |ID  TSN |   SC       VC     TC
------------------+------------------+--------+-----------------------------
12:36:49.335347   | MPUT NT ---------| 4 1BT1 |    1        4      1
12:36:49.336876   | PEND FI ---------| 4  "   |
12:36:49.362784   | WAIT    ---------| 4  "   |
12:36:55.371234   + strt >>>KDCAPPL  + 4  "   +    1        5      1
12:36:55.371897   | INIT    ---------| 4  "   |
    .             |     .            |    .   |               .
    .             |     .            |    .   |               .
12:36:55.372459   | MPUT NT ---------| 4  "   |
12:36:55.372897   | PEND FI ---------| 4  "   |
12:36:55.411213   | WAIT    ---------| 4  "   |
12:36:55.411569   + WAIT    end<<<<< + 4  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<
12:37:03.408459   | strt >>>PERWRT01 | 4 1BT0 |    1        6      1
12:37:03.431278   | INIT    ---------| 4  "   |
    .             |     .            |    .   |               .
    .             |     .            |    .   |               .
12:37:04.095678   + SPUT US ---------+ 4  "   +-----------------------------
12:37:04.095678   |   CC DC 46Z 0000 | 4  "   |    1        6      1
12:37:04.257469   | MPUT NE ---------| 4  "   |
12:37:04.257675   + PEND RE ---------+ 4  "   +-----------------------------
12:37:04.897239   | WAIT    ---------| 4  "   |
12:37:04.897983   | WAIT    end<<<<< | 4  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<
12:37:08.748349   | strt >>>PERWRT02 | 4  "   |    1        6      2
12:37:08.749789   + INIT    ---------+ 4  "   +-----------------------------
    .             |     .            |    .   |               .
    .             |     .            |    .   |               .
12:37:12.034127   + MPUT NE ---------+ 4  "   +-----------------------------
12:37:12.035369   | PEND FI ---------| 4  "   |
12:37:21.682389   | WAIT    end<<<<< | 4  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<
12:37:21.682982   | strt >>>ATAC1016 | 4  "   |    1       11      1
    .             |     .            |    .   |               .
    .             |     .            |    .   |               .
12:37:31.683128   | PEND FI ---------| 4  "   |
12:37:31.685369   + strt >>>ATAC1010 + 1 1BT1 +    1       12      1
12:37:31.706698   | WAIT    ---------| 4 1BT0 |    1       11      1
12:37:31.706887   | WAIT    end<<<<< | 4  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<
12:37:31.706987   | strt >>>ASYNTCCL | 4  "   |    1       13      1
    .             |     .            |    .   |               .
    .             |     .            |    .   |               .
13:23:56.602289   + DBCL USRC      18| 1  "   |
13:23:56.621389   | DBCL USRC       0| 1  "   |
13:23:56.626489   | PEND RE ---------+ 1  "   |
    .             |     .            |    .   |               .
    .             |     .            |    .   |               .
13:23:56.639578   | CONT    ---------| 1  "   |
13:23:56.737897   | DBCL FITA      65| 4  "   |    1       14     30
    .             |     .            |    .   |               .
    .             |     .            |    .   |               .
```

rechter Teil der Liste



14 Anhang

14.1 *open*UTM installieren

Die Installation von *open*UTM erfolgt üblicherweise mit dem Verfahren IMON. *open*UTM wird als Subsystem von BS2000/OSD geladen. Dabei können verschiedene Versionen von *open*UTM als voneinander unabhängige Subsysteme UTM geladen werden.

Wenn Sie die Performance Ihrer Anwendungen mit SM2 oder KDCMON überwachen wollen, müssen Sie neben dem Subsystem UTM das Subsystem UTM-SM2 oder das Subsystem KDCMON laden.

Wenn Sie die Verschlüsselungsfunktionen von *open*UTM nutzen wollen, dann müssen Sie das Zusatzprodukt *open*UTM-CRYPT bestellen und installieren.

Im Folgenden sind spezielle Informationen über *open*UTM, UTM-SM2 und KDCMON zusammengefasst, die für die Installation von Bedeutung sind.

Versionsabhängigkeiten zu anderen Softwareprodukten wie Datenbanksysteme, FHS etc., sowie Angaben zu Compilerversionen und kompatiblen Laufzeitsystemen finden Sie in der Freigabemitteilung. Sie wird als druckaufbereitete Datei auf dem Produktband mit ausgeliefert. Angaben zu Compilerversionen und Laufzeitsystemen finden Sie auch im Abschnitt „Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen“ auf Seite 245.

14.1.1 UTM-Systemcode

Bestandteile des UTM-Systemcodes

Der UTM-Systemcode besteht aus folgenden Teilen:

– dem Basissystem
– dem Abbildungsmodul für die jeweilige BS2000-Version
– den Modulen für UTM-D-Funktionen
– den Modulen für die Verschlüsselungsfunktionen, falls vorhanden

Diese Bestandteile werden beim Laden als getrennte Einheiten behandelt und in der angegebenen Reihenfolge geladen.

Abbildungsmodule - Ablauf von *open*UTM unter verschiedenen BS2000-Versionen

Die Module des UTM-Basissystems sind unabhängig von der Betriebssystemversion. Wenn *open*UTM Funktionen des Betriebssystems braucht, dann ruft *open*UTM diese über „neutrale Schnittstellen“ auf. Die Abbildung der neutralen Schnittstellen auf systemspezifische Schnittstellen erfolgt in einem Abbildungsmodul. Diese Technik erlaubt es, eine *open*UTM-Version in verschiedenen Betriebssystemversionen einzusetzen.

Für jede BS2000-Version, in der *open*UTM ablauffähig ist, gibt es folgendes Abbildungsmodul (= Großmodul):

Abbildungsmodul	BS2000-Version	Plattform
KCYV130	BS2000/OSD V4.0	/390
KCYV140	BS2000/OSD V5.0	/390 + SPARC
KCYV150	BS2000/OSD V6.0	/390 + SPARC

Das zur BS2000-Version gehörende Abbildungsmodul wird aus der Bibliothek dynamisch nachgeladen, in der das UTM-Basissystem liegt. Die Verknüpfung zum Basissystem wird hergestellt.

Namensräume für Module des UTM-Systemcodes

Für die Objektmodule der Bestandteile des UTM-Systemcodes gelten eigene Namensräume. Die Namen für ENTRYs und EXTRNs beginnen mit:

– KCS im Basissystem
– KCY in den Abbildungsmodulen
– KCD in den UTM-D-Modulen
– KCO in XAP-TP-Modulen
– KCE in den Encryption Modulen (Verschlüsselung)

Auslieferung und Liefereinheiten

Alle Bestandteile für die Nutzung der UTM-D Funktionen werden als Teil der Liefereinheit *open*UTM V5.2 ausgeliefert. Für die Nutzung der UTM-D-Funktionen ist jedoch eine Lizenz erforderlich.

Die Verschlüsselungskomponente *open*UTM-CRYPT V5.2 ist aus rechtlichen Gründen eine eigene Liefereinheit und muss getrennt bestellt werden. Hinweise zum Installieren siehe Abschnitt „Encryption Code“ auf Seite 234.

UTM-Systemcode

Der gesamte Systemcode von *open*UTM V5.2 wird in der Bibliothek SYSLNK.UTM.052.TPR (BS2000 auf /390) bzw. SPMLNK.UTM.052.TPR für SX-Maschinen (BS2000 auf SPARC) ausgeliefert:

– Systemcode des Basissystems *open*UTM V5.2A
– Abbildungsmodule für die BS2000-Versionen
– Systemcode der UTM-D-Module

Die Objektkorrekturen für alle Module des UTM-Systemcodes befinden sich in der REP-Datei SYSREP.UTM.052. Die REP-Datei wird bei der Installation mit IMON aus der Lieferdatei SYSRMS.UTM.052 erzeugt. Die Korrekturen werden beim Laden des Subsystems UTM eingefügt.

Die NOREF-Datei SYSNRF.UTM.052 wird mitgeliefert. Die Datei enthält Namen von Modulen, die zur REP-Verarbeitung benötigt werden.

Encryption-Code

Die Verschlüsselungskomponente für *open*UTM befindet sich in der eigenen Liefereinheit *open*UTM-CRYPT V5.2. Der UTM-Systemcode enthält zunächst nur das Modul KCNOCRYP, das als Stellvertreter für den echten Verschlüsselungscode fungiert.

Um die Verschlüsselungsfunktionen nutzen zu können, müssen Sie deshalb das Zusatzprodukt *open*UTM-CRYPT V5.2 bestellen. *open*UTM-CRYPT enthält das Modul KCECRYP mit dem Encryption-Code. KCECRYP müssen Sie *vor* dem Start des UTM-Subsystems in die UTM-Bibliothek kopieren, indem Sie die Prozedur SYSPRC.UTM.052(COPY-CRYPT) aufrufen.

Das Modul KCECRYP wird dann per Autolink aus der Bibliothek mit den UTM-Systemmodulen nachgeladen.

14.1.2 UTM-Systemcode laden

Einträge im Subsystemkatalog für *open*UTM erzeugen

Der UTM-Systemcode läuft als ein Subsystem UTM des Betriebssystems BS2000/OSD ab. Der gesamte UTM-Systemcode inklusive Abbildungsmodul und UTM-D-Modulen wird vom BS2000-Subsystemmanagement (DSSM) geladen.

Der Systemverwalter kann den Subsystemkatalog im laufenden Betrieb verändern. Mit *open*UTM werden SSD-Objekte (Subsystem-Deklaration-Objekte) ausgeliefert. Mit diesen SSD-Objekten kann der Systemverwalter einen Eintrag für das Subsystem UTM im Subsystemkatalog erzeugen.

Folgende SSD-Objekte werden mit *open*UTM V5.2 ausgeliefert:

Datei	Einsatzfall
SYSSSC.UTM.052.130	*open*UTM in BS2000/OSD V4.0 (auf /390)
SYSSSC.UTM.052.140	*open*UTM in BS2000/OSD V5.0 (auf /390 und SPARC)
SYSSSC.UTM.052.150	*open*UTM in BS2000/OSD V6.0 (auf /390 und SPARC)

Die UTM-D-Module sind im Lieferumfang enthalten.

In dem Subsystemkatalog-Eintrag, den der Systemverwalter mit diesen SSD-Objekten erzeugt, sind folgende Voreinstellungen wirksam:

- Ladezeitpunkt:
  Das Subsystem UTM muss explizit mit Kommando START-SUBSYSTEM geladen werden.
- Installation-Userid: *DEFAULT-USERID
- Lade-Bibliothek:
  SYSLNK.UTM.052.TPR auf /390-Anlagen
  SPMLNK.UTM.052.TPR auf SPARC
- REP-Datei: SYSREP.UTM.052

Der Systemverwalter kann diese Voreinstellungen mit dem Kommando CHANGE-SUBSYSTEM ändern.

Laden

Abhängig vom Eintrag im Subsystemkatalog wird das Subsystem UTM wie folgt geladen:

Explizites Laden durch den BS2000-Systemverwalter mit: `/START-SUBSYSTEM SUBSYSTEM-NAME=UTM, VERSION='05.2'`	CREATIM=ONCREA (Standard)
Implizites Laden beim Hochfahren des Systems	CREATIM=AFTSR
Implizites Laden beim ersten Start einer UTM-Anwendung	CREATIM=ONCALL

14.1.3 UTM-Systemcode entladen

Der gesamte UTM-Systemcode, inklusive Abbildungsmodul und *open*UTM-D-Funktionen, kann nur entladen werden, wenn keine UTM-Anwendung mehr läuft. Zum Entladen des Systemcodes von *open*UTM muss der Systemverwalter das folgende Kommando eingeben:

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=UTM, VERSION='05.2'
```

Nach dem Entladen kann der UTM-Systemcode erneut geladen werden, d.h. der UTM-Systemcode kann bei laufendem BS2000-System komplett ausgetauscht werden. Diese Funktion können Sie zum Umsteigen auf eine UTM-Korrekturversion oder auf eine neue *open*UTM-Version nutzen.

Das erneute Laden kann mit Hilfe des Kommandos START-SUBSYSTEM veranlasst werden. Haben Sie im Subsystemkatalog-Eintrag für *open*UTM angegeben, dass der UTM-Systemcode beim ersten Start einer UTM-Anwendung geladen wird, so löst der nächste Start einer UTM-Anwendung das erneute Laden des UTM-Systemcodes aus.

14.1.4 Produktdateien installieren

Für die folgenden UTM-Produktdateien ist im ausgelieferten SSD-Objekt jeweils *DEFAULT-USERID als Installationsname voreingestellt.

```
SYSLNK.UTM.052.TPR
SPMLNK.UTM.052.TPR
SYSREP.UTM.052
```

Der BS2000-Systemverwalter kann beim Installieren der UTM-Produktdateien diese voreingestellten Namen auch ändern. Dabei muss er Folgendes beachten:

- Der Installationsname der REP-Datei muss mit "SYSREP." beginnen, denn IMON erzeugt die REP-Datei beim Installieren aus der RMS-Datei.
- Der Installationsname der NRF-Datei muss gleich dem Namen der REP-Datei sein, nur die Zeichenfolge "SYSREP" wird durch "SYSNRF" ersetzt.

14.1.5 Meldungsdateien

Für die Ausgabe von UTM-Meldungen über die BS2000-Meldungsverarbeitung wird die Meldungsdatei SYSMES.UTM.052 mit *open*UTM ausgeliefert. Bei der Installation mit IMON wird diese Datei automatisch zur BS2000-Meldungsdatei hinzugefügt.

14.1.6 REP-Dateien und RMS-Dateien

Objektkorrekturen werden in Form einer RMS-Datei ausgeliefert. Der Systemverwalter erzeugt daraus bei der Installation direkt mit RMS eine REP-Datei. Für *open*UTM V5.2, UTM-SM2 und KDCMON werden die folgenden RMS-Dateien ausgeliefert:

Subsystem	**RMS-Datei**	**REP-Datei**
UTM	SYSRMS.UTM.052	SYSREP.UTM.052
UTM-SM2	SYSRMS.UTM-SM2.*nnn*	SYSREP.UTM-SM2.*nnn*
KDCMON	SYSRMS.KDCMON.*nnn*	SYSREP.KDCMON.*nnn*

Dabei hängt *nnn* wie folgt von der BS2000-Version ab:

nnn=130 für BS2000/OSD V4.0 (/390)
nnn=140 für BS2000/OSD V5.0 (/390 und SPARC)
nnn=150 für BS2000/OSD V6.0 (/390 und SPARC)

14.1.7 Paralleler Betrieb mehrerer *open*UTM-Versionen

Es ist möglich, im selben BS2000-Betriebssystem mehrere *open*UTM-Versionen nebeneinander zu laden und gleichzeitig im Produktivbetrieb einzusetzen, beispielsweise ist in BS2000/OSD V4.0 der Parallel-Betrieb von *open*UTM 5.1A und *open*UTM V5.2A möglich.

Diese Funktion ist besonders beim Übergang auf eine neue *open*UTM-Version von Vorteil. Sie können dann neben dem Produktivbetrieb einzelne UTM-Anwendungen versuchsweise mit der UTM-Folgeversion ablaufen lassen. Die Umstellung mehrerer UTM-Anwendungen von einer *open*UTM-Version auf eine andere innerhalb eines Rechners kann damit schrittweise getestet und durchgeführt werden.

Beim parallelen Betrieb mehrerer *open*UTM-Versionen müssen Sie Folgendes beachten:

- Im BS2000-Subsystemkatalog sind alle *open*UTM-Versionen einzutragen. Die Einträge müssen den Parallelbetrieb mehrerer *open*UTM-Versionen erlauben. Die mit *open*UTM ausgelieferten SSD-Objekte enthalten diese Erlaubnis.
- Beim Laden des UTM-Systemcodes mit dem Kommando `START-SUBSYSTEM` müssen Sie den Parameter `VERSION-PARALLELISM=COEXISTENCE-MODE` angeben.
- Sämtliche Teile einer Anwendung müssen mit den Dateien/Bibliotheken der gleichen *open*UTM-Version erstellt werden.
- UTM-Anwendungen in verschiedenen *open*UTM-Versionen dürfen nicht den gleichen Namen haben. *open*UTM verhindert den Start der zweiten Anwendung gleichen Namens.

14.1.8 Subsystem UTM-SM2

Der Messmonitor SM2 kann Performancewerte zu laufenden UTM-Anwendungen in der SM2-Messdatei ablegen und am Bildschirm ausgeben. Voraussetzung ist, dass der Baustein UTM-SM2 installiert ist. UTM-SM2 ist in BS2000-GA (Grundausbau) enthalten. UTM-SM2 wird als eigenständiges Subsystem des BS2000 geladen.

Bestandteile

Zu UTM-SM2 gehören die im Folgenden aufgelisteten Bibliotheken und Dateien:

Name	Inhalt
SYSLNK.UTM-SM2.*nnn* SPMLNK.UTM-SM2.*nnn*	Ladebibliothek für /390 Ladebibliothek für SPARC
SYSSSC.UTM-SM2.*nnn*	SSD-Objekt
SYSREP.UTM-SM2.*nnn*	REP-Datei

Dabei bezeichnet *nnn* die UTM-SM2-Version:

nnn=130 für UTM-SM2 V13.0 auf BS2000/OSD V4.0 (/390)
nnn=140 für UTM-SM2 V14.0 auf BS2000/OSD V5.0 (/390 und SPARC)
nnn=150 für UTM-SM2 V15.0 auf BS2000/OSD V6.0 (/390 und SPARC)

Die REP-Datei wird bei der Installation mit IMON aus der Lieferdatei SYSRMS.UTM-SM2.*nnn* erzeugt.

Einträge in den Subsystemkatalog

Mit dem SSD-Objekt kann der Systemverwalter im laufenden Betrieb einen Eintrag für das Subsystem UTM-SM2 im Subsystemkatalog des BS2000 erzeugen.

Bei der Standardinstallation erzeugt IMON für UTM-SM2 automatisch den Eintrag im Subsystemkatalog.

In dem SSD-Objekt SYSSSC.UTM-SM2.*nnn* bzw. SRMSSC.UTM-SM2.*nnn* sind folgende Voreinstellungen wirksam:

- Ladezeitpunkt:
  Das Subsystem UTM-SM2 muss explizit mit Kommando START-SUBSYSTEM geladen werden.
- Installation-Userid:
  *DEFAULT-USERID
- Lade-Bibliothek:
  SYSLNK.UTM-SM2.*nnn* auf /390-Anlagen
  SPMLNK.UTM-SM2.*nnn* auf SPARC
- REP-Datei: SYSREP.UTM-SM2.*nnn*

Der Systemverwalter kann die Voreinstellungen für die Ladebibliothek und für die REP-Datei mit Hilfe des Kommandos `CHANGE-SUBSYSTEM` ändern. Werden die Einstellungen im SSD-Objekt nicht geändert, müssen die Lieferbestandteile in der Kennung $TSOS installiert werden.

Die REP-Datei muss in jedem Fall shareable eingerichtet werden



Die Voreinstellung für den Ladezeitpunkt darf der Systemverwalter **nicht** ändern.

Laden des Subsystems UTM-SM2

UTM-SM2 kann sowohl per Systemverwalter-Kommando als auch durch *open*UTM geladen werden.

- Laden über Kommando:

  ```
  /START-SUBSYSTEM SUBSYSTEM-NAME=UTM-SM2
  ```

- Laden durch *open*UTM, wobei es zwei Möglichkeiten gibt:
  - UTM-SM2 wird automatisch beim Start der Anwendung geladen, falls diese mit MAX SM2=ON generiert ist.
  - UTM-SM2 wird beim Einschalten der Datenlieferung durch die Administration implizit geladen, wenn die Anwendung mit MAX SM2=OFF oder MAX SM2=ON generiert ist.

Entladen von UTM-SM2

Entladen wird UTM-SM2 mit dem Kommando

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=UTM-SM2
```

14.1.9 Subsystem KDCMON

Der UTM-Messmonitor KDCMON ist als eigenständiges Subsystem des BS2000 realisiert. Sie können mit KDCMON UTM-Anwendungen vermessen, die unter verschiedenen *open*UTM-Versionen ablaufen. KDCMON ist in BS2000-GA (Grundausbau) enthalten.

Lieferbestandteile

Zum Einsatz von KDCMON sind die im Folgenden aufgelisteten Bibliotheken, Dateien und Programme erforderlich:

Name	Inhalt
SYSLNK.KDCMON.*nnn* SPMLNK.KDCMON.*nnn*	Ladebibliothek für /390 Ladebibliothek für SPARC
SYSSSC.KDCMON.*nnn*	SSD-Objekt für BS2000/OSD
SYSREP.KDCMON.*nnn*	REP-Datei
SYSMES.KDCMON.*nnn*	Meldungsdatei für BS2000/OSD
SYSPRG.KDCMON.*nnn*	Programm zum Starten von KDCMON
SYSPRG.KDCMON.*nnn*.KDCPMSM	Programm zum Umsetzen und Sortieren der Messdaten

Dabei bezeichnet *nnn* die KDCMON-Version:

nnn=130 für KDCMON V13.0 auf BS2000/OSD V4.0 (/390)
nnn=140 für KDCMON V14.0 auf BS2000/OSD V5.0 (/390 und SPARC)
nnn=150 für KDCMON V15.0 auf BS2000/OSD V6.0 (/390 und SPARC)

Die REP-Datei wird bei der Installation mit IMON aus der Lieferdatei SYSRMS.KDCMON.*nnn* erzeugt.

Einträge in den Subsystemkatalog

Mit dem SSD-Objekt kann der Systemverwalter im laufenden Betrieb einen Eintrag für das Subsystem KDCMON im Subsystemkatalog des BS2000 erzeugen.

Bei der Standardinstallation erzeugt IMON für KDCMON automatisch den Eintrag im Subsystemkatalog.

In dem SSD-Objekt SYSSSC.KDCMON.*nnn* bzw. SRMSSC.KDCMON.*nnn* sind folgende Voreinstellungen wirksam:

- Ladezeitpunkt: KDCMON wird implizit beim ersten Aufruf geladen.
- Installation-Userid:
  *DEFAULT-USERID

Der Systemverwalter kann die Voreinstellungen für die Ladebibliothek und für die REP-Datei mit Hilfe des Kommandos CHANGE-SUBSYSTEM ändern. Werden die Einstellungen im SSD-Objekt nicht geändert, müssen die Lieferbestandteile in der Kennung TSOS installiert werden.

Die REP-Datei muss in jedem Fall shareable eingerichtet werden.

i Die Voreinstellung für den Ladezeitpunkt darf der Systemverwalter **nicht** ändern.

Installieren des Subsystems KDCMON

Die Lieferbestandteile von KDCMON werden beim Installieren unter *DEFAULT-USERID abgelegt.

Die Meldungsdatei SYSMES.KDCMON.*nnn* muss mit dem folgenden Kommando angemeldet werden:

```
/MODIFY-MSG-FILE-ASSIGNMENT ADD-FILE=SYSMES.KDCMON.nnn
```

Laden des Subsystems KDCMON

Um KDCMON zu laden, müssen Sie das Programm SYSPRG.KDCMON.*nnn* unter der TSOS-Kennung starten.

Entladen des Subsystems KDCMON

Das Subsystem KDCMON wird entladen mit dem DSSM-Kommando

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=KDCMON
```

Soll KDCMON entladen werden, wenn die Datenaufzeichnung in einer der UTM-Anwendungen noch nicht abgeschlossen ist, dann muss

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=KDCMON,FORCED=YES
```

angegeben werden.

14.2 UTM-Tools aufrufen

In *open*UTM V5.2 werden die UTM-Tools als LLMs erzeugt und in der Programmbibliothek SYSLNK.UTM.052.UTIL ausgeliefert. Jedes Tool rufen Sie wie folgt auf:

```
/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=$user-id.SYSLNK.UTM.052.UTIL,-
ELEMENT=toolname)
```

toolname ist der Name des Tools, z.B. KDCDEF, und $*user-id* die Installations-Benutzerkennung (IMON-Installationspfad).

Kompatibilitätsmodus

Aus Kompatibilitätsgründen können die UTM-Tools auch noch über die bisherige Schnittstelle aufgerufen werden:

```
/START-PROGRAM FROM-FILE=$user-id.SYSPRG.UTM.052.toolname
```

SYSPRG.UTM.052.*toolname* enthält jedoch nur einen Verweis auf die Programmbibliothek SYSLNK.UTM.052.UTIL, die standardmäßig im IMON-Installationspfad gesucht wird.

Falls sich SYSLNK.UTM.052.UTIL nicht im IMON-Installationspfad befindet, z.B. weil eine Korrekturversion getestet wird, dann muss vor Aufruf des Tools die Job-Variable JV-UTM-LLM-*toolname* eingerichtet und zugewiesen werden. Dazu wird die Prozedur SET-JV-UTIL in der Bibliothek SYSPRC.UTM.052 ausgeliefert. Diese Prozedur richtet die Job-Variablen für alle UTM-Tools ein und weist die Nachladebibliothek zu.

14.3 Speicherverbrauch einer UTM-Anwendung

Der Speicherbedarf auf der Festplatte beträgt ca. 23.000 PAM-Seiten für alle Produktdateien.

In der folgenden Tabelle ist zusammengefasst, welche Bestandteile beim Betrieb von UTM-Anwendungen Adressraum in Speicherklasse 4, 5 oder 6 benötigen:

Bestandteil	**Klasse**	**Größe**	
Systemcode *open*UTM V5.2	4	3088 KB	1.
Systemcode *open*UTM-CRYPT V5.2	4	124 KB	2.
statische KAA-Tabellen	5		3.
dynamische KAA-Tabellen	5		4.
Cache-Speicher	5		5.
KTA-Tabellen	5		6.
KDCROOT-Code-Module	6	440 KB	
Zusatzbedarf bei Formatierung	6		7.
Zusatzbedarf bei DB-System	6		8.
Administrationsprogramm KDCADM	6	276 KB	
Teilprogramme des Anwenders	6		9.

1. Systemcode *open*UTM:
   Der UTM-Systemcode wird durch das BS2000-Subsystemmanagement DSSM in den Klasse-4-Speicher geladen. Zum UTM-Systemcode gehören die Komponenten UTM-Basis inklusive Betriebssystem-Anpassmodul und der UTM-D-Code inclusive XAP-TP. Einzelheiten dazu finden Sie im Abschnitt „openUTM installieren“ auf Seite 231.

2. Der UTM-Verschlüsselungscode wird durch das DSSM ebenfalls in den Klasse-4-Speicher geladen. Sehen Sie dazu Abschnitt „UTM-Systemcode“ auf Seite 234.

3. Statische KAA-Tabellen (**K**DC **A**pplication **A**rea):
   Mit der Meldung `K450 KDCFILE generated; KAA-size:nnnK;...` protokolliert KDCDEF bei der Generierung, dass die statischen KAA-Tabellen *nnn* KB belegen. Beim Start der Anwendung liest *open*UTM diese Tabellen in einen Memory Pool im Klasse-5-Speicher ein. Der Memory Pool belegt *nnn* KB, aufgerundet auf ein Vielfaches von 1 MB.

   Die Hinzunahme eines Benutzers und eines Terminals verursacht einen Mehrbedarf von (mindestens) 920 byte. Für „Extras“ wie Kellerung, TLS, ULS etc. steigt dieser Wert. Für eine Abschätzung des Platzbedarfs durch Erweiterung der Konfiguration ist es besser, die Anzahl der Objekte mit RESERVE-Anweisungen in den KDCDEF-Input einzutragen und KDCDEF den Platzbedarf selbst ausrechnen zu lassen.

4. Dynamische KAA-Tabellen:
   Der Platzbedarf für ein Terminal ist mindestens 1 KB. *open*UTM nutzt zunächst den Verschnitt am Ende des Memory Pools, der für die statischen KAA-Tabellen angelegt wurde. Wenn dieser Platz nicht ausreicht, legt *open*UTM einen oder mehrere weitere Memory Pools für die dynamischen Tabellen im Klasse-5-Speicher an.

5. Cache-Speicher:
   Als Pufferbereich für I/O's zum Pagepool auf KDCFILE benützen alle Prozesse einer UTM-Anwendung den Cache-Speicher. *open*UTM legt ihn als Memory Pool im Klasse-5-Speicher an. Seine Größe wird durch die KDCDEF-Parameter CACHESIZE=(*number*,...) und BLKSIZE=*blocksize* bestimmt.
   Bei *blocksize*=2K ist er *number* * 2KB groß, aufgerundet auf ein Vielfaches von 1 MB.

6. KTA-Tabellen (**K**DC **T**ask **A**rea):
   Der Bereich enthält prozessspezifische Verwaltungsdaten, er wird im Klasse-5-Speicher angelegt. Die Länge ist mindestens 8KB. Der Platzbedarf steigt in Abhängigkeit von den KDCDEF-Generierungsparametern.

7. Zusatzbedarf bei Formatierung:
   Der Code des Formatierungssystems wird nachgeladen. Zum Speicherbedarf für das Formatierungssystem siehe Angaben von FHS. Dazu kommt der Speicherbedarf für die Formate.

8. Zusatzbedarf bei Datenbanksystem:
   Das Verbindungsmodul des Datenbanksystems kann statisch oder dynamisch dazugebunden werden. Den Speicherbedarf entnehmen Sie den Informationen des jeweiligen Datenbanksystems.

9. Teilprogramme des Anwenders:
   Den Speicherbedarf entnehmen Sie der Binderliste.

14.4 Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen

In diesem Abschnitt finden Sie Informationen dazu, welche Versionen der Compiler und Laufzeitsysteme der einzelnen Programmiersprachen Sie zum Erzeugen vom UTM-Teilprogrammen einsetzen können. Für jede Programmiersprache, die *open*UTM unterstützt, sind in einer Tabelle folgende Sachverhalte aufgelistet:

- Versionen des Compilers, die Sie benutzen können, um die Objekte (OM oder LLM) eines UTM-Teilprogramms zu erzeugen
- Versionen des Laufzeitsystems, die für diese Teilprogramme geeignet sind
- Werte für den Operanden COMP der Steueranweisung PROGRAM, die Sie bei der KDCDEF-Generierung angeben müssen, um diese Teilprogramme in die Anwendungskonfiguration aufzunehmen

Den Tabellen können Sie insbesondere entnehmen, welche Kombinationen von Compiler-Version, Version des Laufzeitsystems und COMP-Wert erlaubt sind.

Weitere Informationen über zulässige Compiler- und Laufzeitsystem-Versionen finden Sie auch im Handbuch zu CRTE und in den Unterlagen zu den Compilern (Freigabemitteilung, Handbücher).

Beachten Sie, dass im Folgenden auch Compiler-Versionen aufgeführt sind, für die die Wartung bereits eingestellt ist. Der Grund dafür ist, dass in einigen Kunden-Anwendungen ältere Programme als Objekte vorhanden sind und weiterhin benützt werden sollen.

Für alle Weiter- und Neuentwicklungen sollten Sie grundsätzlich die Compiler- und Laufzeitsystem-Versionen einsetzen, die noch gewartet werden! Andernfalls haben Sie bei Problemen keinen Anspruch auf Gewährleistung oder Korrekturen.

Generell sind die Hinweise und Einschränkungen in den Freigabemitteilungen und Handbüchern der Compiler zu beachten.

i *open*UTM V5.2 setzt eine CRTE-Version ≥ V2.3 voraus. Es wird empfohlen, mindestens diese CRTE-Version einzusetzen. Außerdem wird dringend empfohlen, bei Compilern und Laufzeitsystemen immer einen aktuellen Korrekturstand einzusetzen

Sprachmix in einer Anwendung

Sprachmix bedeutet, dass eine UTM-Anwendung aus Teilprogrammen zusammengesetzt werden darf, die mit den Compilern unterschiedlicher Sprachen erzeugt wurden und die im Ablauf entsprechend unterschiedliche Laufzeitsysteme benutzen. Dabei ist jedoch Folgendes zu beachten:

Eine UTM-Anwendung darf für **jede** Programmiersprache, die von *open*UTM unterstützt wird und mit PROGRAM ... COMP≠ILCS generiert werden muss, nur **ein** Laufzeitsystem enthalten.

Sprachmix innerhalb eines Teilprogramms

*open*UTM erlaubt einen Sprachmix auch innerhalb eines einzelnen Teilprogramms, d.h. Teilprogramme können aus mehreren Quellcodes zusammengesetzt sein, die in unterschiedlichen Programmiersprachen geschrieben sind. Voraussetzung dafür ist, dass die Programmteile sich nach den gleichen Verknüpfungskonventionen aufrufen. Dabei ist anzustreben, dass sich die Programmteile nach den ILCS-Konventionen aufrufen, siehe CRTE-Handbuch. Insbesondere müssen die Datendarstellungen bei der Übergabe von Parametern und beim Zugriff auf gemeinsame Datenstrukturen identisch sein.



ACHTUNG!
Die Hinweise in Abschnitt „Hinweise für Anwendungen mit ILCS-Teilprogrammen“ auf Seite 54 sind unbedingt einzuhalten.

Beim Aufruf von Assembler-Unterprogrammen sind folgende Fälle zu unterscheiden:

- Sowohl das aufrufende Programm als auch das gerufene Unterprogramm sind ILCS-Programme. In diesem Fall sind keine Einschränkungen zu beachten.

  Es ist möglich, Assembler-ILCS-Programme mit Z-Makros zu programmieren. Siehe dazu *open*UTM-Handbuch „Anwendungen programmieren mit KDCS für Assembler“ (auf *open*UTM Handbuch-CD).

- Aufrufendes und gerufenes Programm sind keine ILCS-Programme, halten aber die gleichen Verknüpfungskonventionen ein. Wie diese Veknüpfungskonventionen aussehen, ist in den Benutzerhandbüchern der jeweiligen Programmiersprache beschrieben.

Durch unbeabsichtigte Sprachübergänge bei den folgenden Konstruktionen können Fehler in der Sprachverknüpfung oder Adressfehler auftreten:

– Aufruf von C-Funktionen
– Aufruf von Assembler-Funktionen mit @-Makros
– Aufruf von Fremdsoftware
– Aufruf von Datenbanken
– Aufruf von Formatierungssystemen

14.4.1 Assembler

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
ASSEMBH V1.1 bis V1.2	—	ASSEMB [1]
ASSEMBH V1.2 [2]	ASSEMBH V1.2 [2]	ILCS [2]

Anmerkungen zur Tabelle

1. Wenn Sie COMP=ASSEMB angeben, dann dürfen Sie das ASSEMBH-Laufzeitsystem **nicht** benutzen. Der Grund dafür ist, dass das Laufzeitsystem von ASSEMBH ab Version V1.1 das ILCS nutzt. Es kommt dann zu einer verbotenen Mischung von Nicht-ILCS- mit ILCS-Programmteilen.
2. Der Compiler und das Laufzeitsystem müssen mindestens den Korrekturstand V1.2A10 haben und das Assembler-Programm **muss** ILCS-fähig sein.
   Es gibt zwei Möglichkeiten, ein Assembler-Programm ILCS-fähig zu machen:
   - Sie benutzen die Assembler-Makros ZSTRT, ZCALL und ZEND in der Ausprägung ZSTRT ILCS=YES. Bitte beachten Sie, dass die Ausprägung ZSTRT ILCS=NO (nicht-ILCS-fähig) Standardwert ist!
   - Sie benutzen die Makros @ENTR ... ILCS=YES..., @PASS und @EXIT.

14.4.2 C/C++

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
C/C++ V3.0	CRTE ab V2.3 [1]	ILCS [2]

Anmerkungen zur Tabelle

1. *open*UTM V5.2 setzt eine CRTE-Version ≥ V2.3 voraus. Es wird empfohlen, mindestens diese CRTE-Version einzusetzen. Außerdem wird dringend empfohlen, bei Compilern und Laufzeitsystemen immer einen aktuellen Korrekturstand einzusetzen.
2. Sie können auch COMP=C setzen. Dieser Wert wird von KDCDEF aber ohne Warnung mit COMP=ILCS überschrieben.

14.4.3 Cobol

Compiler	Laufzeitsystem	PROGRAM ..., COMP=
COBOL85 V2.3A03	CRTE ab V2.3[1]	ILCS / COB1 [2]
COBOL2000 V1.0	CRTE ab V2.3[1]	ILCS / COB1 [2]

Anmerkungen zur Tabelle

1. *open*UTM V5.2 setzt eine CRTE-Version ≥ V2.3 voraus. Es wird empfohlen, mindestens diese CRTE-Version einzusetzen. Außerdem wird dringend empfohlen, bei Compilern und Laufzeitsystemen immer einen aktuellen Korrekturstand einzusetzen.
2. Die von COBOL85 oder COBOL2000 erzeugten Objekte sind gleichermaßen für ILCS-Verknüpfung und Nicht-ILCS-Verknüpfung geeignet; es gibt keine Compiler-Option, um gezielt ILCS- oder Nicht-ILCS-Objekte zu erzeugen.

   COMP=ILCS bedeutet, dass die ILCS-Verknüpfung verwendet wird. COMP=COB1 bedeutet Nicht-ILCS-Verknüpfung (= Verknüpfung alter Art).

   Der Wert von COMP= wird nach folgenden Kriterien gesetzt:

   – COMP=ILCS wird angegeben, wenn alle aufgerufenen Unterprogramme ebenfalls ILCS-fähig sind.

   – COMP=COB1 muss immer dann angegeben werden, wenn der Cobol-Modul Unterprogramme aufruft, die nicht ILCS-fähig sind.

Compiler-Optionen

Die folgende Tabelle gibt einen Überblick darüber, welche COMOPT-Parameter in Abhängigkeit vom verwendeten Compiler gesetzt werden müssen und was dabei zu beachten ist.

COMOPT-Parameter	Compiler	Bemerkung
CHECK-CALLING-HIERARCHY=NO	COBOL85 COBOL2000	Nur bei COMP=COB1, d.h. bei Nicht-ILCS-Verknüpfung.
MARK-LAST-PARAMETER=YES	COBOL2000	Empfehlung

Mischen von Cobol-Programmen

Wenn ein Teilprogramm mit COMP=COB1 generiert ist, dann darf es aus Modulen bestehen, die mit dem COB1-, COBOL85- oder COBOL2000-Compiler übersetzt wurden. Wenn das Teilprogramm mit COMP=ILCS generiert ist, dann darf es nur aus Modulen bestehen, die mit dem COBOL85- oder COBOL2000-Compiler übersetzt wurden.

Innerhalb einer Anwendung können Teilprogramme, die mit COMP=ILCS generiert sind, und Teilprogramme, die mit COMP=COB1 generiert sind, nebeneinander existieren.

14.4.4 Fortran

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
FOR1 V2.2 [1]	FOR1 V2.2	ILCS
Fortran90 V1.0 [2]	Fortran90 V1.0	ILCS

Anmerkungen zur Tabelle

1. Für den Compiler FOR1 V2.2 wird die ILCS-Fähigkeit über die Compiler-Option eingestellt:
   - Wenn das Teilprogramm übersetzt wurde mit der Compiler-Option COMOPT LINKAGE=FOR1-SPECIFIC, dann muss in der PROGRAM-Anweisung COMP=FOR1 angegeben werden. Diese Compiler-Option legt fest, dass das Programm die „alte“ FOR1-Schnittstelle nutzt.
   - Wenn das Teilprogramm mit der Compiler-Option COMOPT LINKAGE=STD (Standardwert) übersetzt wurde, dann muss in der PROGRAM-Anweisung COMP=ILCS angegeben werden. Diese Compiler-Option legt fest, dass das Programm die ILCS-Schnittstelle verwendet.
2. Die Fortran90-Teilprogramme müssen mit der folgenden Compiler-Option übersetzt werden:

```
COMPILER-ACTION = OBJECT-GENERATION( ... , LINKAGE = ILCS )
```

Bemerkung zum Sprachmix

In einer UTM-Anwendung dürfen FOR1-Teilprogramme enthalten sein, die mit unterschiedlichen Werte für PROGRAM ...,COMP= in die Konfiguration eingetragen wurden. In einer UTM-Anwendung können somit nebeneinander Teilprogramme existieren, von denen eines die „alte“ FOR1-Schnittstelle und ein anderes die ILCS-Schnittstelle nutzt.

14.4.5 Pascal

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
Pascal-XT ab V2.2B	V2.2	ILCS

14.4.6 PL/I

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
PLI1 V4.2	[1]	PLI1
PLI1 ab V4.2A [2]	[1]	ILCS

Anmerkungen

1. Beachten Sie bitte, dass die Compilerobjekte die gleiche Version wie das benutzte Laufzeitsystem haben müssen. Sie dürfen nicht mehrere PLI1-Laufzeitsysteme in eine UTM-Anwendung einbinden.
2. Mit PLI1 einer Version ≥ V4.2 können auch ILCS-Teilprogramme erzeugt werden. Dazu müssen Sie bei der PROCEDURE-Anweisung OPTIONS(ILCS) angeben. Sie müssen dann darauf achten, dass Ihre UTM-Anwendung nur PL/I-Teilprogramme enthält, die ILCS-fähig sind.

14.4.7 SPL4

Compiler	Laufzeitsystem	PROGRAM ...,COMP=
SPL4 ab V2.0A	[1]	SPL4 / ILCS [2]

Anmerkungen zur Tabelle

1. Mit *open*UTM wird derzeit das Laufzeitsystem SPL V2.7A ausgeliefert. Dies kann sich jedoch mit einer Korrekturausgabe ändern. Lesen Sie dazu bitte die aktuelle Freigabemitteilung.
2. Die ILCS-Fähigkeit des Teilprogramms hängt von der Compiler-Option ab:
   - Wird das Teilprogramm mit der Compiler-Option GEN=(ILCS=NO) übersetzt (Nicht-ILCS-Verknüpfung), dann muss es mit COMP=SPL4 generiert werden.
   - Wird das Teilprogramm mit der Compiler-Option GEN=(ILCS=YES) übersetzt (ILCS-Verknüpfung), dann muss es mit COMP=ILCS generiert werden.

14.4.8 Hinweise für die Umstellung von einer älteren *open*UTM-Version

Bei der Umstellung einer UTM-Anwendung von einer *open*UTM-Version < V4.0 auf die *open*UTM-Version 5.2 ist u.a. Folgendes zu beachten:

– In *open*UTM-Version < V4.0 ist in der PROGRAM-Anweisung der Standardwert COMP=COB1 wirksam.

– In *open*UTM V5.2 hat der Parameter COMP= **keinen** Standardwert, d.h. man muss ihn immer angeben. Der bisherige Standardwert COMP=COB1 kann weiterhin angegeben werden. Der Wert COMP=COBOL85 ist **nicht** mehr erlaubt.

Um zu klären, welcher Wert COMP=... für ein Teilprogramm geeignet ist, prüfen Sie bitte Folgendes:

- Mit welchem Compiler ist das Teilprogramm übersetzt worden? Welche Version des Compilers wurde verwendet? Welche Compiler-Optionen wurden dabei gesetzt? Ist demnach ein ILCS-fähiges Objekt entstanden?

  Zur Klärung lesen Sie bitte die Information zu den einzelnen Compilern in den Abschnitten ab Seite 247.

- Ruft das Teilprogramm Unterprogramme auf? Sind diese Unterprogramme ILCS-fähig?

  Wenn sowohl das Teilprogramm als auch alle eventuellen Unterprogramme ILCS-fähig sind, dann sollte das Teilprogramm mit COMP=ILCS generiert werden.

  Wenn das Teilprogramm oder eines der Unterprogramme nicht ILCS-fähig ist oder wenn sich dies nicht klären lässt, dann gehen Sie wie folgt vor:

  – Bei Nicht-Cobol-Teilprogrammen behalten Sie zunächst den bisherigen Wert COMP=... bei.

  – Bei Cobol-Teilprogrammen setzen Sie COMP=COB1, auch dann, wenn das Programm mit dem Compiler COBOL85 übersetzt wurde. Beim Binden des UTM-Anwendungsprogramms müssen Sie das bisherige Cobol-RTS bereitstellen.



Es wird daher **dringend empfohlen**, bei einer Umstellung einer älteren UTM-Anwendung auf die ILCS-Verknüpfung überzugehen. Die ILCS-Verknüpfung hat den Vorteil, dass man einerseits Programme verschiedener Programmiersprachen problemlos untereinander aufrufen kann und dass andererseits die Wartung und Weiterentwicklung der Compiler und Laufzeitsysteme gesichert ist.

14.5 Aufbau der Accountingsätze von *open*UTM

Die Accountingsätze von *open*UTM werden in die BS2000 Accounting-Datei geschrieben und mit RAV ausgewertet. Die Sätze entsprechen den Konventionen für BS2000-Accounting-Sätze. Deshalb sind die ersten 24 Byte für das BS2000-Accounting reserviert, die restlichen Daten enthalten UTM-spezifische Informationen.

Es gibt folgende zwei Satztypen:

– Abrechnungssätze (Satztyp UTMA)

– Kalkulationssätze (Satztyp UTMK)

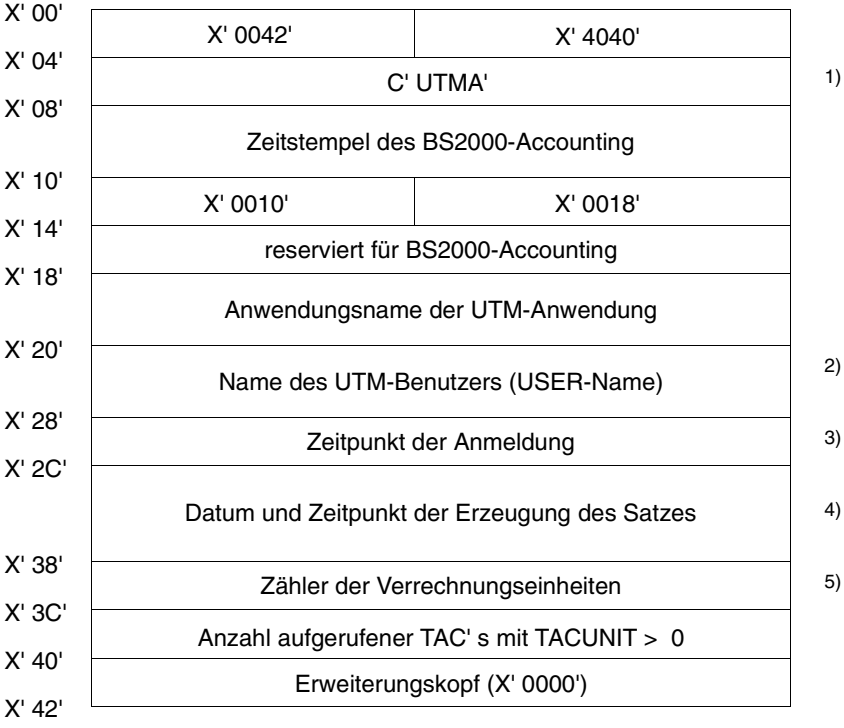
Die Bedeutung dieser Sätze ist im Kapitel „Accounting“ auf Seite 177 beschrieben.

Zum Aufbau von Abrechnungs- und Kalkulationssatz in der BS2000-Abrechnungsdatei benötigen Sie das BS2000-Makro ARDS.

Wird das ARDS mit UTM=NEW aufgerufen, dann greift ARDS auf die UTM- Makros KDCUTMA und KDCUTMK zu. KDCUTMA und KDCUTMK stehen in der UTM-Makrobibliothek SYSLIB.UTM.052.ASS. Sie beschreiben die Struktur von Abrechnungssatz bzw. Kalkulationssatz.

Wird ARDS mit UTM=OLD aufgerufen, dann werden die BS2000-eigenen Makros ARDSUTMA und ARDSUTMK verwendet.

14.5.1 Aufbau des Abrechnungssatzes



Die Anmerkungen bedeuten:

1) Kennung des Abrechnungssatzes beim BS2000-Accounting

2) Name des Benutzers. In einer UTM-Anwendung ohne generierte Benutzer trägt *open*UTM den Namen des LTERM-Partners ein.

3) Zeitpunkt der Anmeldung dieses Benutzers (USER) an diesem LTERM in Sekunden, bezogen auf die Zeitbasis der BS2000-Version.
Wenn im aktuellen Lauf der UTM-Anwendung für diesen USER nur Asynchron-TACs aufgerufen wurden, ist der Inhalt dieses Feldes binär Null.

4) Format: *yymmddhhmmss* (Jahr/Monat/Tag/Stunde/Minute/Sekunde)

5) Summe der Verrechnungseinheiten für diesen Benutzer, seit der letzte Verrechnungssatz geschrieben wurde bzw. seit dem Zeitpunkt der Anmeldung.

14.5.2 Aufbau des Kalkulationssatzes

X' 00'	X' 005E'	X' 4040'	
X' 04'	C' UTMK'		1)
X' 08'	Zeitstempel des BS2000-Accounting		
X' 10'	X' 0010'	X' 0034'	
X' 14'	reserviert für BS2000-Accounting		
X' 18'	Anwendungsname der UTM-Anwendung		
x' 20'	Transaktionscode des Teilprogramms		
X' 28'	CPU-Zeit in *open*UTM (msec)		
X' 2C'	CPU-Zeit im Datenbanksystem (msec)		
X' 30'	Anzahl I/O ' s in *open*UTM		
X' 34'	Anzahl I/O ' s im Datenbanksystem		
X' 38'	Länge der Eingabenachricht		
X' 3C'	Länge der Ausgabenachricht		
X' 40'	Anzahl der asynchronen Ausgaben		
X' 44'	Verrechnungseinheiten für LTAC's		2)
X' 48'	Name des UTM-Benutzers		
X' 50'	Name des LTERM-Partners		
X' 58'	Realzeit des Teilprogrammlaufs (in msec)		
X' 5C'	Erweiterungskopf (X' 0000')		
X' 5E'			

Die Anmerkungen bedeuten:

1) Kennung des Satzes beim BS2000-Accounting

2) siehe KDCDEF-Anweisung LTAC...,LTACUNIT=

14.6 Aufbau der SAT-Protokollsätze

Sicherheitsrelevante UTM-Ereignisse können mit der BS2000-Funktion SAT (**S**ecurity **A**udit **T**rail) protokolliert werden. Die Protokollierung von UTM-Ereignissen durch SAT ermöglicht die Beweissicherung, die nach den F2/Q3-Kriterien des ITS-Katalogs gefordert wird. Bei eingeschalteter SAT-Protokollierung wird eine Mindestprotokollierung durchgeführt. Zusätzlich können weitere Ereignisse definiert werden. Die Protokollierung dieser Ereignisse in den SAT-Protokollsätzen kann ereignisspezifisch, benutzerspezifisch und auftragsspezifisch ein- und ausgeschaltet werden.

14.6.1 Bedeutung der von *open*UTM benutzten Protokolldatenfelder

Für jedes Ereignis erzeugt *open*UTM einen SAT-Protokollsatz. Jeder Protokollsatz, den *open*UTM an SAT übergibt, besteht aus einem Teil fester Struktur und Länge, dem SAT-Header, dem ein Teil variabler Struktur und Länge folgt.

Der SAT-Header enthält Datum und Uhrzeit, den BS2000-Benutzer, die TSN, das aktuelle BS2000-Ereignis und dessen Resultat. Dabei werden nur die folgenden Felder von *open*UTM versorgt:

Feldname	Bedeutung	Typ
EVT	Typ des Ereignisses	C-string(1..3) bei *open*UTM immer „TRM“
RES	Ergebnis des Ereignisses	Set: Success/Failure („S“ / „F“)

Der variable Teil enthält einzelne UTM-spezifische Datenfelder, denen jeweils ein Längenfeld und ein SAT-Identifier vorangeht. Die Art und Anzahl der einzelnen Datenfelder hängt vom Typ des UTM-Ereignisses ab.

Die folgenden Tabellen enthalten die von *open*UTM benutzten Protokolldatenfelder, die Bedeutung des Feldinhalts und dessen Datentyp.

UTM-Ereignisse können mit der ALARM-Funktion von SAT gekoppelt werden. Der Datentyp für SAT-ALARM stimmt bis auf wenige Ausnahmen mit dem Datentyp für SATUT überein. In der Tabelle auf Seite 257 ist der ALARM-Datentyp in Klammern () angegeben, wenn er sich vom Datentyp für SATUT unterscheidet.

Feldname	Bedeutung	Typ
APPLNAM	BCAM-Anwendungsname	C-String (1..8)
COMMAND	Name UTM-SAT-Administrationskommando oder Administrations-Programmschnittstelle	C-String (1..8)
DATNAM1	Name UTM-Speicherbereich	C-String (1..8)
DATNAM2	Name UTM-Objekt	C-String (1..8)
DATTYPE	Typ des UTM-Speicherbereichs	Set: G / T / U
LTERM	LTERM-Partnername für Clients und Drucker	C-String (1..8)
MUXLTRM	MUX-Transportverbindung	C-String (1..8)
PTERM	BCAM-Name	C-String (1..8)
TACNAID	Transaktionskennzeichen	Set: G/T/D/C/P
TACNAM	Transaktionscode	C-String (1..8)
USER2	UTM-Benutzername	C-String (1..8)
UTMACTY	Zugriffsart UTM-Speicherbereich	Set: C / D / READ / WRITE
UTMAPPL	UTM-Anwendungsname	C-String (1..8)
UTMCALL	Adresse des Aufrufers	X-String (1..4) (ALARM: X-String 4..4)
UTMHEX3	RC der Administrations-Programmschnittstelle	C-String (1..8)
UTMNAME	Name Lademodul oder Administrations-Programmschnittstelle: Name UTM-Objekt	C-String (1..32)
UTMOBJ1	Name UTM-Objekt oder Administrations-Programmschnittstelle: Name des modifizierten Feldes	C-String (1..8)
UTMOBJ2	Name UTM-Objekt oder Administrations-Programmschnittstelle: Inhalt des modifizierten Feldes	C-String (1..8)
UTMOBJ3	Administrations-Programmschnittstelle: Objekttyp	C-String (1..8)
UTMOBJ4	Programm-Lademodus oder Administrations-Programmschnittstelle: Subopcode1 oder „FORMATTR“ oder Feldname	C-String (1..8) (ALARM: C-String 8..8)
UTMOBJ5	Administrations-Programmschnittstelle: Format-Attribut oder Feldname	C-String (1..8)
UTMOBJ6	Lademodul-Version	C-String (1..24) (ALARM: C-String 64..64)
UTMREAS	KDCS-Returncode [1] Administrations-Programmschnittstelle: leer	C-String (1..8)

Feldname	Bedeutung	Typ
UTMSTAT	Transaktionsstatus	C-String (1)
UTMSUBC	UTM-Ereignis	Set: CHG-ACC-KEY / CHANGE-PW / SIGN / DATA-ACCESS / ADM-CMD / START-PU/ END-PU / TASK-ON/ TASK-OFF / SEL-CMD/ CHG-PROG
UTMTAID	Transaktionsidentifikation [2]	X-String (1..4) (ALARM: X-String 4..4)
UTMUSER	UTM-Benutzername	C-String (1..8)

[1] Der Returncode setzt sich bei den UTM-Ereignissen CHANGE-PW, START-PU, END-PU und DATA-ACCESS zusammen aus dem kompatiblen und dem inkompatiblen KDCS-Returncode.

[2] Die Transaktionsidentifikation (TA-ID) setzt sich zusammen aus zwei Byte Vorgangszähler (innerhalb einer Session) und zwei Byte Transaktionszähler (innerhalb eines Vorgangs). Sie wird an der Schnittstelle IUTMDB zur Datenbank für die SAT-Protokollierung der Datenbank-Ereignisse bereitgestellt. Die Transaktionsidentifikation dient dazu, ein Ereignis der Transaktion zuzuordnen, die es erzeugt hat.

14.6.2 Versorgung der Datenfelder

In diesem Abschnitt wird beschrieben, welche Protokolldatenfelder bei der Protokollierung der einzelnen Ereignisse versorgt werden.

Die folgenden Tabellen geben einen Überblick darüber, welche Protokolldatenfelder in Abhängigkeit von den Ereignissen versorgt werden. Die Protokolldatenfelder sind in der Reihenfolge aufgeführt, in der sie im SAT-Protokollsatz erscheinen. Danach wird die Bedeutung der Feldinhalte für die einzelnen Ereignisse aufgeführt.

Bedeutung der Einträge in den folgenden Tabellen:

—	Feld wird nicht versorgt
Y	Feld wird versorgt (Mandatory-Felder im Sinne von SAT)
O	Feld wird in bestimmten Fällen versorgt

Feldname	UTM-Ereignisse										
	CHG-ACC-KEY	TASK-ON	TASK-OFF	SIGN	CHANGE-PW	START-PU	END-PU	DATA-ACCESS	ADM-CMD	SEL-CMD	CHG-PROG
APPLNAM	-	-	-	O	-	-	-	-	O	-	-
UTMUSER	O	-	-	Y	Y	Y	Y	Y	Y	Y	-
UTMAPPL	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
UTMSUBC	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LTERM	-	-	-	Y	-	O	Y	-	O	-	-
PTERM	-	-	-	Y	-	-	-	-	O	-	-
MUXLTRM	-	-	-	O	-	-	-	-	O	-	-
DATNAM1	-	-	-	-	-	-	-	Y	-	-	-
DATNAM2	-	-	-	-	-	-	-	O	O	O	-
DATTYPE	-	-	-	-	-	-	-	Y	-	-	-
UTMACTY	-	-	-	-	-	-	-	Y	-	-	-
COMMAND	-	-	-	-	-	-	-	-	Y	Y	-
UTMOBJ1	-	Y	Y	Y	Y	Y	Y	-	O	O	-
UTMOBJ2	-	-	-	-	-	O	-	-	O	O	-
UTMOBJ3	-	-	-	-	-	-	-	-	O	-	-
UTMCALL	Y	-	-	-	-	-	-	-	-	-	-
TACNAM	O	-	-	-	Y	O	Y	Y	Y	Y	-
TACNAID	-	-	-	-	-	Y	Y	-	-	-	-
USER2	-	-	-	-	-	-	-	-	O	O	-
UTMNAME	-	-	-	-	-	-	-	-	O	-	Y
UTMTAID	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y
UTMSTAT	-	-	-	-	-	-	Y	-	-	-	-
UTMREAS	-	-	-	Y	Y	Y	Y	Y	Y	Y	-
UTMOBJ4	-	-	-	-	-	-	-	-	O	-	Y
UMTOBJ5	-	-	-	-	-	-	-	-	O	-	-
UTMOBJ6	-	-	-	-	-	-	-	-	O	-	Y
UTMHEX3	-	-	-	-	-	-	-	-	Y	-	-

Im Folgenden ist der Aufbau des variablen Teils des Protokollsatzes für jedes Ereignis detailliert beschrieben.

CHG-ACC-KEY: Speicherzugriffsschlüssel umschalten

UTMUSER	UTM-Benutzername. Das Feld wird nur versorgt, falls ein UTM-Benutzer das Ereignis ausgelöst hat.
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	CHG-ACC-KEY
UTMCALL	Adresse des Aufrufers
TACNAM	Transaktionscode. Das Feld wird versorgt, falls der Lauf eines Teilprogramms das Ereignis ausgelöst hat.

TASK-ON: Anschließen einer Task an die UTM-Anwendung

UTMAPPL	Name der laufenden Anwendung
UTMSUBC	TASK-ON
UTMOBJ1	F für die erste (first) Task oder N für eine Folgetask (next Task) oder L wird beim Programmaustausch oder beim PEND ER ausgelöst (Load Program)

TASK-OFF: Abmelden einer Task von der UTM-Anwendung

Es wird nur die normale Beendigung einer Task protokolliert.

UTMAPPL	Name der laufenden Anwendung
UTMSUBC	TASK-OFF
UTMOBJ1	Letzte Task: ja "Y" oder nein "N"

SIGN: Anmelden eines UTM-Benutzers

APPLNAM	BCAM-Anwendungsname
UTMUSER	Name des Benutzers/Clients, der das Ereignis auslöst
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	SIGN
LTERM	Name des LTERM-Partners, über den sich Benutzer/Clients anschließen

PTERM	BCAM-Name der Benutzer/Clients, die definierten LTERM-Partnern zugeordnet sind
MUXLTRM	PTERM-Name der MUX-Transportverbindung
UTMTAID	Transaktionsidentifikation oder null
UTMOBJ1	BCAM-Prozessorname
UTMREAS	Rückgabefeld KCRSIGN

CHANGE-PW: Passwortänderung

CHANGE-PW wird auch ausgelöst, wenn ein Benutzerpasswort vom UTM-Administrator geändert wird.

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöste (eventuell Administratorkennung)
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	CHANGE-PW
UTMOBJ1	Name der Benutzerkennung, deren Passwort geändert wird
UTMREAS	KDCS-Returncodes
TACNAM	Transaktionscode des laufenden Teilprogramms
UTMTAID	Transaktionsidentifikation oder null

START-PU: Erzeugen eines Auftrags bzw. Start eines Teilprogramms

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöst
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	START-PU (start program unit)
LTERM	Name des definierten LTERM-Partners oder leer
UTMOBJ1	Bei TACNAID=G: TAC des Auftrags Bei TACNAID=C/T/P: TAC des laufenden Vorgangs
UTMOBJ2	Bei TACNAID=G: DPUT-Identifikation des erzeugten Auftrags
UTMREAS	KDCS-Returncodes (bei TACNAID=G)
UTMTAID	Transaktionsidentifikation der laufenden Transaktion oder null
TACNAM	TAC des laufenden Teilprogramms

TACNAID Transaktionskennzeichen. Mögliche Werte (die Werte G, C, T, P schließen sich dabei gegenseitig aus):

- G für Auftrag erzeugt (generate)
  Dialogaufträge werden nur als erzeugt protokolliert, wenn sie wegen TAC-Klassensteuerung nicht sofort gestartet werden können
- C für Vorgangsbeginn (begin conversation)
  Mit diesem Teilprogramm beginnt ein Vorgang
- T für Transaktionsbeginn (begin transaction)
  Mit diesem Teilprogramm beginnt eine Folgetransaktion eines Vorgangs
- P für Beginn eines Folgeteilprogramms innerhalb einer Transaktion (begin program run)

Das Erzeugen von Nachrichten an einen LTERM-Partner wird nicht protokolliert.
Ein Quittungsauftrag (an einen TAC) wird erst dann als erzeugt protokolliert, wenn er auf Grund des Ergebnisses des Hauptauftragslaufes ausgewählt und in einen Hauptauftrag umgewandelt wird. Das Ziel des abgelaufenen Hauptauftrags wird dabei im Feld LTERM bzw. TACNAM ausgegeben.

END-PU: Beendigung eines Teilprogramms

UTMUSER Name des UTM-Benutzers, der das Ereignis auslöst

UTMAPPL Name der laufenden Anwendung

UTMSUBC END-PU (end program unit)

LTERM Name des definierten LTERM-Partners oder leer

UTMOBJ1 Transaktionscode des laufenden Vorgangs

UTMSTAT Das Feld wird nur bei TACNAID=T oder C versorgt. Es enthält dann den Transaktionsstatus:

- C Transaktionssicherung (Commit)
- R Rücksetzen der Transaktion (Rollback)

UTMTAID Transaktionsidentifikation

TACNAM Transaktionscode des laufenden Teilprogramms

TACNAID Mögliche Werte:

- C Ende des Teilprogramms und des Vorgangs (end of conversation)
- T Ende des Teilprogramms und der Transaktion; der Vorgang wird fortgesetzt (end of transaction)
- P Ende des Teilprogramms; die Transaktion wird fortgesetzt (end of program run)

UTMREAS KDCS-Returncodes

DATA-ACCESS: Zugriff auf einen UTM-Speicherbereich

UTMUSER Name des UTM-Benutzers, der das Ereignis auslöst

UTMAPPL Name der laufenden Anwendung

UTMSUBC DATA-ACCESS

DATNAM1 Name des angesprochenen UTM-Speicherbereiches

DATNAM2 Bei DATTYPE=ULS: UTM-Benutzer,
bei DATTYPE=TLS: LTERM-Partner für Clients und Drucker
sonst leer

DATTYPE Type des Speicherbereichs:

- G GSSB
- U ULS
- T TLS

UTMACTY Art des Speicherzugriffs:

- READ lesend
- WRITE schreibend
- C erzeugt (create)
- D gelöscht (delete)

UTMREAS KDCS-Returncodes

UTMTAID Transaktionsidentifikation

TACNAM Transaktionscode des laufenden Teilprogramms

ADM-CMD: Aufruf der Administrations-Programmschnittstelle

UTMUSER	Name des UTM-Benutzers, der das Ereignis auslöst
UTMAPPL	Name der laufenden Anwendung
UTMSUBC	ADM-CMD
COMMAND	DADM (KDCS-Aufruf) oder Aufruf der Programmschnittstelle zur Administration: CHNGAPPL / CREATE / CREASTMT / DELETE / DUMP / GETOBJ / MODIFY / SENDMSG / SHUTDOWN / SPOOLOUT / SYSLOG / USLOG
UTMREAS	Bei COMMAND=DADM KDCS-Returncodes Administrations-Programmschnittstelle: leer
UTMHEX3	Returncode der Administrations-Programmschnittstelle
UTMTAID	Transaktionsidentifikation
TACNAM	Transaktionscode des laufenden Teilprogramms
USER2	Session, Benutzer oder leer

Die Felder DATNAM2, LTERM und UTMOBJ1 werden eventuell leer ausgegeben.

i Bei manchen Funktionsaufrufen der Programmschnittstelle sind verschiedene Aktionen möglich. In diesem Fall schreibt SAT für jede Aktion einen Protokollsatz und protokolliert den Parameter, wobei nur für geänderte Parameter auch der Parameterwert ausgegeben wird.

Sehen Sie auch *open*UTM-Handbuch „Anwendungen administrieren“.

In Abhängigkeit von COMMAND werden zusätzlich folgende Felder versorgt:

Bei CHNGAPPL

UTMOBJ4	Subopcode1 (NEW/OLD bei PROGRAM)

Bei CREATE

UTMOBJ3	Objekt-Typ
UTMNAME	Objekt-Name

Bei DELETE

UTMOBJ3	Objekt-Typ
UTMOBJ4	Subopcode1 (NEW/OLD bei PROGRAM)
UTMNAME	Objekt-Name

Bei MODIFY

UTMOBJ3 Objekt-Typ
In Abhängigkeit des Objekt-Typs werden weitere Felder protokolliert:

Objekt-Typ	**protokollierte Felder**
LOADMODU	UTMOBJ1: „Version“ UTMOBJ6: Version UTMNAME: Name des zu modifizierenden Lademoduls
LPAP	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden LPAP-Partners
LSES	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden LSES- und CON-Tripels
LTAC	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden LTACs
LTERM	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden LTERM-Partners
MUX	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden MUX-Tripels
OSICON	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden OSI-CON
OSILPAP	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden OSI-LPAP-Partners
PTERM	PTERM: Name des zu modifizierenden Clients/Drucker (PRTM) UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden PTERM-Tripels
TAC	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name des zu modifizierenden TACs
TACCLASS	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMNAME: Name der zu modifizierenden TACCLASS

Objekt-Typ	protokollierte Felder
TPOOL	LTERM: LTERM-Präfix UTMOBJ1: Parameter1 UTMOBJ2: Parameterwert1 UTMOBJ4: Parameter2 UTMOBJ5: Parameterwert2 UTMNAME: Name des zu modifizierenden TPOOLs (LTERM-Präfix, PRONAM, PTYPE, BCAMAPPL)
USER	UTMOBJ1: Parameter UTMOBJ2: Parameterwert UTMOBJ4: „FORMATTR“ UTMOBJ5: Formatattribut UTMNAME: Name des zu modifizierenden USERs
MAXPAR	UTMOBJ1: Parameter UTMOBJ2: Parameterwert
TIMERPAR	UTMOBJ1: Parameter UTMOBJ2: Parameterwert
TASKSPAR	UTMOBJ1: Parameter UTMOBJ2: Parameterwert
DIAGACCP	UTMOBJ1: Parameter UTMOBJ2: Parameterwert

Bei SENDMSG

LTERM LTERM-Name oder „KDCALL“

Bei SHUTDOWN

UTMOBJ1 Parameter

UTMOBJ2 Parameterwert

UTMOBJ4 Subbopcode1

Bei SPOOLOUT

UTMOBJ1 „SPOOLOUT“

UTMOBJ2 „ON“

Bei SYSLOG

UTMOBJ1 Subbopcode1

Bei USLOG

UTMOBJ1 Subbopcode1

Bei DADM (KDCS-Aufruf):

(Nur das Löschen wird protokolliert.)

DATNAM2 Ziel (TAC) der zu löschenden Aufträge (Das Löschen von Nachrichten an einen Client oder Drucker wird nicht protokolliert.)

UTMOBJ1 DPUT-ID des zu löschenden Auftrags oder KDCALL (alle Aufträge eines Zieles werden gelöscht)

USER2 Auftraggeber des zu löschenden Auftrags (bei UTMOBJ1=KDCALL leer)

UTMOBJ2 Wert des KCMOD-Parameters beim DADM-Aufruf (bei UTMOBJ1=KDCALL leer)

SEL-CMD: Ausführung eines Preselection-Kommandos

Ist der Name oder Wert eines Parameters länger als 8 Zeichen, so wird die bei der UTM-SAT-Administration beschriebene Abkürzung im Protokolldatenfeld ausgegeben.

UTMUSER Name des UTM-Benutzers, der das Ereignis auslöst

UTMAPPL Name der laufenden Anwendung

UTMSUBC SEL-CMD

COMMAND Angabe für das UTM-SAT-Administrationskommando: MSATSEL oder MSATPROT

UTMREAS Systeminterner Returncode

UTMTAID Transaktionsidentifikation

TACNAM Transaktionscode des laufenden Teilprogramms

USER2 Benutzer oder leer

Die Felder DATNAM2, LTERM und UTMOBJ1 werden eventuell leer ausgegeben.

In Abhängigkeit von COMMAND werden zusätzlich folgende Felder versorgt:

MSATSEL (Preselection steuern, Kommando KDCMSAT SATSEL=...)

Für jeden angegebenen Namen wird ein eigener Protokolldatensatz geschrieben. Es wird immer nur genau eines der Felder USER2, DATNAM2 oder UTMOBJ1 versorgt:

USER2	UTM-Benutzer als Objekt zur Preselection
DATNAM2	TAC als Objekt zur Preselection
UTMOBJ1	Ereignis als Objekt zur Preselection
UTMOBJ2	Preselection-Wert (NONE, SUCC, FAIL, BOTH oder OFF)

MSATPROT (SAT-Protokollierung steuern, Kommando KDCMSAT SAT=...)

UTMOBJ2	Protokollierung ein- (ON) bzw. ausgeschaltet (OFF)

CHG-PROG: Austausch eines Lademoduls

UTMAPPL	Name der laufenden UTM-Anwendung
UTMSUBC	CHG-PROG
UTMNAME	Name des auszutauschenden Moduls
UTMOBJ4	Lade-Modus des auszutauschenden Moduls
UTMOBJ6	Neue Modulversion
UTMTAID	Transaktionsidentifikation, falls UTMOBJ4=ON-CALL, sonst null



Für UTMOBJ4=ON-CALL ist zu beachten, dass CHG-PROG immer als Erfolg protokolliert wird. Ob der Austausch erfolgreich war, kann am zugehörigen END-PU-Protokolldatensatz abgelesen werden. (Im Fehlerfall kommt ein PEND ER mit entsprechenden KDCS-Returncodes.) Auch das erstmalige Laden wird als Austausch protokolliert.

14.7 Beispielprogramme

Für *open*UTM werden standardmäßig Beispielprogramme ausgeliefert, die dem Benutzer Teilaspekte der Anwendungserstellung erleichtern sollen. Nachfolgend werden einige dieser Beispielprogramme näher erläutert. Die Beschreibung der Beispielprogramme für die Administration finden Sie im *open*UTM-Handbuch „Anwendungen administrieren“.

14.7.1 Beispielprogramme zum Anmeldevorgang

Mit den Beispielprogrammen zum Anmeldevorgang kann ein einfacher Anmeldevorgang mit FHS-Formatausgabe realisiert werden.

Es werden sowohl übersetzte Objekte als auch Quellprogramme (Cobol) ausgeliefert. Sie haben damit die Möglichkeit, die neuen Funktionen ohne Programmierung schnell auszuprobieren. Mit den Quellprogrammen haben Sie eine Programmiervorlage für den Fall, dass Sie einen Anmeldevorgang nach eigenen Wünschen realisieren möchten.

Funktionen

Die Programme sind für alle Generierungsvarianten geeignet.

Der Terminalbenutzer wird nach Verbindungsaufbau (wenn erforderlich) zur Eingabe seiner Berechtigungsdaten aufgefordert. Zu diesem Zweck gibt der Anmeldevorgang ein "Begrüßungsformat" aus, das zwei Eingabefelder für Benutzerkennung und Passwort enthält. Das Passwort ist abdruckbar anzugeben. Bitte beachten Sie bei der Angabe von Benutzerkennung und Passwort, dass Kleinbuchstaben in Großbuchstaben umgesetzt werden.

Wenn *open*UTM die Berechtigungsdaten ablehnt, dann wiederholt der Anmeldevorgang die Eingabe-Aufforderung. Er gibt dazu das gleiche Format aus, es enthält nun einen Hinweis auf die Ablehnung. Nach drei Fehlversuchen wird der Anmeldevorgang abgebrochen.

Akzeptiert *open*UTM die Berechtigungsdaten, verhält sich der Anmeldevorgang wie folgt:

- Kein Vorgangswiederanlauf:
  Ist ein Startformat generiert, so wird es ausgegeben, andernfalls eine Aufforderung zur Eingabe im Zeilenmodus.
- Vorgangswiederanlauf:
  Der Bildschirmwiederanlauf des offenen Vorgangs wird veranlasst.

Am Terminal werden englische Texte ausgegeben. Auch die Kommentare im Quellprogramm sind englisch.

Bestandteile

Die Beispielprogramme zum Anmeldevorgang finden Sie in der Bibliothek SYSLIB.UTM.052.EXAMPLE.

Element	LMS-Typ	Bedeutung
SIGN1	S	Cobol-Source → 1. Teilprogramm
SIGN2	S	Cobol-Source → 2. Teilprogramm
KDCSIGN1	R	Bindemodul → 1. Teilprogramm
KDCSIGN2	R	Bindemodul → 2. Teilprogramm
FORSIGN	R	Bildschirmformat (' *' -Format)
FORSIGN	S	Adressierungshilfe
FORSIGN	F	IFG-Source

Integration in eine UTM-Anwendung

Um den Beispiel-Anmeldevorgang in eine UTM-Anwendung zu integrieren, sind die KDCDEF-Generierungsanweisungen wie folgt zu erweitern:

```
PROGRAM KDCSIGN1,COMP=ILCS
PROGRAM KDCSIGN2,COMP=ILCS
TAC   KDCSGNTC,PROGRAM=KDCSIGN1
TAC   TACSIGN2,PROGRAM=KDCSIGN2,CALL=NEXT
```

Der TAC-Name TACSIGN2 ist einprogrammiert. Er ist am Anfang der beiden Teilprogramme jeweils als Konstante definiert und dadurch bei Bedarf leicht änderbar.

Der SPAB muss mindestens 600 byte, der KB-Programmbereich mindestens 2 Byte lang sein (siehe MAX-Anweisung im *open*UTM-Handbuch „Anwendungen generieren“).

Die Binder-Anweisung ist wie folgt zu erweitern:

```
INCLUDE-MODULES LIBRARY=$userid.SYSLIB.UTM.052.EXAMPLE
               ,ELEMENT=(KDCSIGN1, KDCSIGN2)
```

Das Bildschirmformat FORSIGN ist in die Formatbibliothek der Anwendung aufzunehmen. Das Format ist für Terminals des Typs 8160, 9750, 9755 und 9763 geeignet.

Das Teilprogramm SIGN1 verwendet die COBOL85-spezifische Anweisung EVALUATE.

14.7.2 CPI-C-Beispielprogramme

CPI-C-Beispielprogramme finden Sie in der Bibliothek SYSLIB.UTM.052.XOPEN.

Element	LMS-Typ	Bedeutung	
KCPSAM1.C	S	C-Source	→ Asynchroner Teil
KCPSAM2.C	S	C-Source	→ Synchroner Teil
KCPSAM1#	R	Objektmodul	→ Asynchroner Teil
KCPSAM2#	R	Objektmodul	→ Synchroner Teil
KCPSAM1@	R	Objektmodul	→ Asynchroner Teil
KCPSAM2@	R	Objektmodul	→ Synchroner Teil

14.7.3 Beispielprogramme zur Asynchronverarbeitung für UPIC-Clients

Für die Asynchronverarbeitung für UPIC-Clients werden mit *open*UTM die drei Teilprogramme UPDIAG, UPASYN und UPDGET ausgeliefert.

Funktionen

Mit diesen drei Teilprogrammen soll wird gezeigt, wie man von einem UPIC-Client aus asynchrone Aufträge erteilen und asynchron über das Ergebnis informiert werden kann.

In diesem Beispiel wird die Asynchronnachricht zunächst an die USER-Queue der Benutzerkennung geschickt, unter der sich der UPIC-Client angemeldet hat. Anschließend wird die Nachricht von einem Dialogteilprogramm gelesen und am Client ausgegeben. Der Vorteil asynchroner Verarbeitung liegt z.B. darin, dass der Benutzer am UPIC-Client sofort nach der Auftragsannahme einen neuen Auftrag eingeben kann und nicht bis zur Auftragsbeendigung blockiert ist.

Diese drei Programme haben folgende Funktionen:

- UPDIAG liest eine Eingabenachricht, schickt diese als Asynchronauftrag an das Teilprogramm UPASYN und gibt eine Auftragsannahmebestätigung am Client aus.
- UPASYN empfängt diese Nachricht, wartet 5 Sekunden zur Simulation einer komplexen Verarbeitung und schreibt das Ergebnis in die USER-Queue der Benutzerkennung, unter der sich der UPIC-Client angemeldet hat.

- UPDGET liest mit Warten (60 Sekunden) aus der USER-Queue dieser Benutzerkennung. Falls eine Nachricht vorliegt, wird diese mit MPUT an den UPIC-Client gesendet und mit PEND RE der eigene TAC nochmals aufgerufen, um auf die nächste Nachricht zu warten. Falls keine Nachricht vorliegt, wird UPDGET erneut gestartet, sobald eine Nachricht eintrifft oder die Wartezeit abgelaufen ist, ohne dass eine Nachricht eingetroffen ist.

Bestandteile

Die Quellprogramme und Objektmodule werden als Elemente der Bibliothek SYSLIB.UTM.052.EXAMPLE ausgeliefert. Die Beispielprogramme sind nur in Verbindung mit einem darauf abgestimmten UPIC-Clientprogramm sinnvoll einsetzbar.

Integration in eine UTM-Anwendung

Der UPIC-Client hält zwei Verbindungen zur UTM-Anwendung und meldet sich jeweils mit derselben Benutzerkennung bei *open*UTM an. Diese Benutzerkennung muss mit RESTART=NO generiert werden. Der Client startet im ersten Thread den Dialogvorgang UPDGET zum Lesen der Asynchronnachrichten. Im zweiten Thread startet der Client auf explizite Anforderung den Dialogvorgang UPDIAL, der dann einen Asynchronauftrag an UPASYN erzeugt.

14.7.4 Beispielprogramme für die XML-Schnittstelle

Mit *open*UTM werden die beiden XML-Teilprogramme KXLCTP (XML-Teilprogramm in C) und KXLCOBTP (XML-Teilprogramm in COBOL) ausgeliefert.

Mit diesen Teilprogrammen kann ein XML-Dokument erzeugt, gelesen und geändert werden. Die Funktionalität der Programme, die zum Aufruf notwendigen Transaktionscodes sowie die notwendigen KDCDEF-Anweisungen sind in den Sourcen als Kommentar beschrieben.

Die Quellprogramme und Objektmodule werden als Elemente der Bibliothek SYSLIB.UTM.052.EXAMPLE ausgeliefert.

14.8 Beispielprozeduren

Mit *open*UTM werden standardmäßig Beispielprozeduren ausgeliefert, die Ihnen die Arbeit mit *open*UTM erleichtern sollen. Die Prozeduren sind als Werkzeuge und Vorlage gedacht, die Sie nach Ihren Wünschen modifizieren und erweitern können. Die Prozeduren enthalten teilweise englische Kommentare.

Die Bibliothek SYSPRC.UTM.052 enthält folgende Prozeduren:

Prozedur	**Funktion**
BTRACE	Mischen von BCAM-Trace-Ausgabedateien und auswerten mit dem Programm KDCBTRC
COPY-CRYPT	Encryption-Modul (Verschlüsselungs-Code) in UTM-Systemcode-Bibliothek kopieren
DUMP	UTM-Dumps auswerten
FGGUSLOG	Benutzerprotokoll-Datei als Dateigenerationsgruppe (FGG) einrichten und auf neue Benutzer-Protokolldatei-Generation umschalten
GEN	UTM-Anwendung generieren
LINK	UTM-Anwendungsprogramm binden
MSGMOD	Benutzereigene (Meldungs-)module erzeugen
PAMSAM	Von KDCMON aufgezeichnete Daten aufbereiten und sortieren
SET-JV-UTIL	Job-Variable für Dienstprogramme setzen
SLOG-FGG	Einzelne Dateigenerationen einer SYSLOG-FGG auswerten
SHOW-ETPND	ETPND eines UTM-Moduls anzeigen
START	UTM-Produktivanwendung starten (keine BLS-Anwendung)
START-BLS-APPLICATION	UTM-Produktivanwendung mit BLS starten
SYSLOG	SYSLOG-Datei aufbereiten
UPD	Daten in KDCFILE übertragen mit KDCUPD

14.9 XS-Unterstützung von UTM-Anwendungen

In diesem Abschnitt sind einige Besonderheiten aufgelistet, die Sie beim Binden und Starten von UTM-Anwendungen berücksichtigen müssen, die in den oberen Adressraum einer XS-Anlage geladen werden und im 31-Bit-Modus ablaufen sollen. Dabei werden Kenntnisse der XS-Programmierung und zum XS-Einsatz vorausgesetzt. Nähere Informationen dazu finden Sie im Handbuch „Einführung in die XS-Programmierung“.

*open*UTM unterstützt nur Anwendungen, die:

- entweder vollständig XS-fähig sind und im 31-Bit-Modus ablaufen oder
- nicht XS-fähig ganz im unteren Adressraum (d.h. unterhalb 16 Mbyte) geladen sind und im 24-Bit-Modus ablaufen

Eine UTM-Anwendung kann also nur dann im oberen Adressraum (>16Mbyte) geladen werden und im 31-Bit-Modus ablaufen, wenn alle Bestandteile des UTM-Anwendungsprogramms XS-fähig sind.



ACHTUNG!
UTM-Anwendungen im „mixed mode“ (d.h. mit Umschaltung zwischen 24- und 31-Bit-Adressmodus) werden von *open*UTM nicht unterstützt. Das bedeutet, dass *open*UTM nicht für den ordnungsgemäßen Ablauf einer UTM-Anwendung garantiert, wenn z.B. ein Teilprogramm, das im 31-Bit-Modus abläuft, selbstständig Module im 24-Bit-Modus nachlädt und beim Übergang jeweils den Adressmodus umschaltet.

Übersetzen und Binden

Beim Übersetzen und Binden einer XS-fähigen UTM-Anwendung sind folgende Regeln zu berücksichtigen:

- Alle Teilprogramme müssen mit den Attributen AMODE=ANY (Adressierungsmodus) und RMODE = ANY (Residenzmodus) übersetzt werden.
- Zum Binden einer XS-fähigen UTM-Anwendung müssen Sie den BINDER verwenden.

  Beim Binden der UTM-Anwendung überprüft der Binder die Attribute AMODE und RMODE für alle Teilprogramme und legt für das erzeugte Bindemodul der UTM-Anwendung einen Pseudo-RMODE fest. Dabei orientiert er sich an der „schwächsten“ Komponente, d.h. das Modul erhält nur dann das Attribut RMODE=ANY, wenn alle Komponenten das Attribut RMODE=ANY haben. Wurde eine Komponente mit RMODE=24 übersetzt, dann wird dem Modul RMODE=24 zugeordnet.

  Das Attribut AMODE wird durch den Programmabschnitt (CSECT) bestimmt, der die Einsprungstelle des Bindemoduls enthält.
  Nähere Informationen finden Sie im BS2000-Handbuch „Bindelader-Starter“.

Besonderheiten beim Start einer UTM-Anwendung

Ob die UTM-Anwendung in den oberen oder unteren Adressraum geladen wird, ist abhängig vom UTM-Anwendungsprogramm selbst und vom Wert des Parameters PROGMODE, den Sie beim Aufruf des START-PROGRAM-Kommandos angeben.

- Bei PROGMODE=24 wird die Anwendung in den unteren Adressraum geladen und der 24-Bit-Modus wird eingestellt.
- Bei PROGMODE=ANY:
  Ob die Anwendung in den oberen oder unteren Adressraum geladen wird und welcher Adressierungsmodus eingestellt wird, ist abhängig von den Attributen AMODE und RMODE des Lademoduls (siehe Abschnitt „XS-Unterstützung von UTM-Anwendungen“ auf Seite 273).

  Wenn das Binder-Lader-System (BLS) beim Starten der UTM-Anwendung feststellt, dass alle Bestandteile der UTM-Anwendung, die beim Start der Anwendung geladen werden, XS-fähig sind, dann wird die UTM-Anwendung in den oberen Adressraum geladen.

Soll *open*UTM in der Startphase einer in den oberen Adressraum geladenen Anwendung ein nicht XS-fähiges Modul nachladen, dann bricht *open*UTM den Startvorgang mit einer entsprechenden Fehlermeldung ab.

Zu beachten ist auch, dass von einem Anwendungsprogramm, das im oberen Adressraum läuft, im laufenden Betrieb kein 24-Bit-Modul (ONCALL) nachgeladen werden darf.

Eine UTM-Anwendung kann auch mit /EXEC im oberen Adressraum geladen werden.

Um sicherzustellen, dass eine nicht-XS-fähige UTM-Anwendung in den unteren Adressraum geladen wird und im 24-Bit-Modus abläuft, müssen Sie beim Binden der UTM-Anwendung eine TRAITS-Anweisung mit AMODE=24 einfügen.

Speicherbelegung von UTM-Anwendungen

Die anwendungsspezifischen Tabellen und Datenbereiche (KAA, KTA, Slots und UTM-Cache legt *open*UTM im Klasse 5 Speicher im oberen Adressraum an. UTM-Anwendungen, die im unteren Adressraum laufen, wird dadurch kein Adressraum weggenommen.

Tools KDCDEF, KDCDUMP und KDCUPD

Die UTM-Tools KDCDEF, KDCDUMP und KDCUPD laufen nur im oberen Adressraum (> 16MByte) ab.

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.

Ablaufinvariantes Programm
reentrant program
siehe *reentrant-fähiges Programm*.

Abnormale Beendigung einer UTM-Anwendung
abnormal termination of a UTM application
Beendigung einer *UTM-Anwendung*, bei der die *KDCFILE* nicht mehr aktualisiert wird. Eine abnormale Beendigung wird ausgelöst durch einen schwerwiegenden Fehler, z.B. Rechnerausfall, Fehler in der Systemsoftware. Wird die Anwendung erneut gestartet, führt *open*UTM einen *Warmstart* durch.

abstrakte Syntax (OSI)
abstract syntax
Eine abstrakte Syntax ist die Menge der formal beschriebenen Datentypen, die zwischen Anwendungen über *OSI TP* ausgetauscht werden sollen. Eine abstrakte Syntax ist unabhängig von der eingesetzten Hardware und der jeweiligen Programmiersprache.

Access-List
access list
Eine Access-List definiert die Berechtigung für den Zugriff auf einen bestimmten *Service*, auf eine bestimmte *TAC-Queue* oder auf eine bestimmte *USER-Queue*. Eine Access-List ist als *Keyset* definiert und enthält einen oder mehrere *Keycodes*, die jeweils eine Rolle in der Anwendung repräsentieren. Benutzer oder LTERMs dürfen nur dann auf den Service oder die *TAC-Queue*/*USER-Queue* zugreifen, wenn ihnen die entsprechenden Rollen zugeteilt wurden, d.h. wenn ihr *Keyset* und die Access-List mindestens einen gemeinsamen *Keycode* enthalten.

Access Point (OSI)
siehe *Dienstzugriffspunkt.*

ACID-Eigenschaften
ACID properties

Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Administration
administration

Verwaltung und Steuerung einer *UTM-Anwendung* durch einen *Administrator* oder ein *Administrationsprogramm*.

Administrationskommando
administration command

Kommandos, mit denen der *Administrator* einer *UTM-Anwendung* Administrationsfunktionen für diese Anwendung durchführt. Die Administrationskommandos sind als *Transaktionscodes* realisiert.

Administrationsprogramm
administration program

Teilprogramm, das Aufrufe der *Programmschnittstelle für die Administration* enthält. Dies kann das Standard-Administrationsprogramm *KDCADM* sein, das mit *open*UTM ausgeliefert wird, oder ein vom Anwender selbst erstelltes Programm.

Administrator
administrator

Benutzer mit Administrationsberechtigung.

AES

AES (Advanced Encryption Standard) ist der aktuelle symmetrische Verschlüsselungsstandard, festgelegt vom NIST (National Institute of Standards and Technology), basierend auf dem an der Universität Leuven (B) entwickelten Rijndael-Algorithmus. Wird das AES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen AES-Schlüssel.

Akzeptor (CPI-C)
acceptor

Die Kommunikationspartner einer *Conversation* werden *Initiator* und Akzeptor genannt. Der Akzeptor nimmt die vom Initiator eingeleitete Conversation mit Accept_Conversation entgegen.

Anmeldevorgang (KDCS)
sign-on service

Spezieller *Dialog-Vorgang*, bei dem die Anmeldung eines Clients an eine UTM-Anwendung durch *Teilprogramme* gesteuert wird.

Anschlussprogramm
linkage program
siehe *KDCROOT*.

Anwendungsinformation
application information
Sie stellt die Gesamtmenge der von der *UTM-Anwendung* benutzten Daten dar. Dabei handelt es sich um Speicherbereiche und Nachrichten der UTM-Anwendung, einschließlich der aktuell auf dem Bildschirm angezeigten Daten. Arbeitet die UTM-Anwendung koordiniert mit einem Datenbanksystem, so gehören die in der Datenbank gespeicherten Daten ebenfalls zur Anwendungsinformation.

Anwendungsprogramm
application program
Ein Anwendungsprogramm bildet den Hauptbestandteil einer *UTM-Anwendung*. Es besteht aus der Main Routine *KDCROOT* und den *Teilprogrammen*. Es bearbeitet alle Aufträge, die an eine *UTM-Anwendung* gerichtet werden.

Anwendungswiederanlauf
application restart
Start einer *UTM-S*-Anwendung nach einer vorhergehenden abnormalen Beendigung. Dabei wird die *Anwendungsinformation* auf den zuletzt erreichten konsistenten Zustand gesetzt. Unterbrochene *Dialog-Vorgänge* werden dabei auf den zuletzt erreichten *Sicherungspunkt* zurückgesetzt, so dass die Verarbeitung an dieser Stelle wieder konsistent aufgenommen werden kann (*Vorgangswiederanlauf*). Unterbrochene *Asynchron-Vorgänge* werden zurückgesetzt und neu gestartet oder beim zuletzt erreichten *Sicherungspunkt* fortgesetzt. Bei UTM-F-Anwendungen werden beim Start nach einer vorhergehenden abnormalen Beendigung lediglich die dynamisch geänderten Konfigurationsdaten auf den zuletzt erreichten konsistenten Zustand gesetzt.

Application Context (OSI)
application context
Der Application Context ist die Menge der Regeln, die für die Kommunikation zwischen zwei Anwendungen gelten sollen. Dazu gehören z.B. die abstrakten Syntaxen und die zugeordneten *Transfer-Syntaxen*.

Application Entity (OSI)
application entity
Eine Application Entity (AE) repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen global (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title* (AET). Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Title (OSI)
application entity title
: Ein Application Entity Title ist ein global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title* des jeweiligen *Application Process* und dem *Application Entity Qualifier*.

Application Entity Qualifier (OSI)
application entity qualifier
: Bestandteil des *Application Entity Titles*. Der Application Entity Qualifier identifiziert einen *Dienstzugriffspunkt* innerhalb der Anwendung. Ein Application Entity Qualifier kann unterschiedlich aufgebaut sein. *open*UTM unterstützt den Typ „Zahl“.

Application Process (OSI)
application process
: Der Application Process repräsentiert im *OSI-Referenzmodell* eine Anwendung. Er wird durch den *Application Process Title* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (OSI)
application process title
: Gemäß der OSI-Norm dient der Application Process Title (APT) zur global (d.h. weltweit) eindeutigen Identifizierung von Anwendungen. Er kann unterschiedlich aufgebaut sein. *open*UTM unterstützt den Typ *Object Identifier*.

Application Service Element (OSI)
application service element
: Ein Application Service Element (ASE) repräsentiert eine Funktionsgruppe der Anwendungsschicht (Schicht 7) des *OSI-Referenzmodells*.

Association (OSI)
association
: Eine Association ist eine Kommunikationsbeziehung zwischen zwei *Application Entities*. Dem Begriff Association entspricht der *LU6.1*-Begriff *Session*.

Asynchron-Auftrag
queued job

Auftrag, der vom Auftraggeber zeitlich entkoppelt durchgeführt wird. Zur Bearbeitung von Asynchron-Aufträgen sind in *open*UTM *Message Queuing* Funktionen integriert, vgl. *UTM-gesteuerte Queue* und *Service-gesteuerte Queue*. Ein Asynchron-Auftrag wird durch die *Asynchron-Nachricht*, den Empfänger und ggf. den gewünschten Ausführungszeitpunkt beschrieben.
Ist der Empfänger ein Terminal oder Drucker, so ist der Asynchronauftrag ein *Ausgabeauftrag*; ist der Empfänger ein Asynchron-Vorgang derselben oder einer fernen Anwendung, so handelt es sich um einen *Hintergrundauftrag*.
Asynchron-Aufträge können *zeitgesteuerte Aufträge* sein oder auch in einen *Auftragskomplex* integriert sein.

Asynchron-Conversation
asynchronous conversation

CPI-C-Conversation, bei der nur der *Initiator* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein asynchroner Transaktionscode generiert sein.

Asynchron-Nachricht
queued message

Asynchron-Nachrichten sind Nachrichten, die an eine *Message Queue* gerichtet sind. Sie werden von der lokalen *UTM-Anwendung* zunächst zwischengespeichert und dann unabhängig vom Auftraggeber weiter verarbeitet. Je nach Empfänger unterscheidet man folgende Typen von Asynchron-Nachrichten:

– Bei Asynchron-Nachrichten an eine *UTM-gesteuerte Queue* wird die Weiterverarbeitung komplett durch *open*UTM gesteuert. Zu diesem Typ gehören Nachrichten, die einen lokalen oder fernen *Asynchron-Vorgang* starten (vgl. auch *Hintergrundauftrag*) und Nachrichten, die zur Ausgabe an ein Terminal oder einen Drucker geschickt wird (vgl. auch *Ausgabeauftrag*).
– Bei Asynchron-Nachrichten an eine *Service-gesteuerte Queue* wird die Weiterverarbeitung durch einen *Service* der Anwendung gesteuert. Zu diesem Typ gehören Nachrichten an eine *TAC-Queue*, Nachrichten an eine *USER-Queue* und Nachrichten an eine *Temporäre Queue*. Die User-Queue und die Temporäre Queue müssen dabei zur lokalen Anwendung gehören, die TAC-Queue kann sowohl in der lokalen als auch in einer fernen Anwendung liegen.

Asynchron-Programm
asynchronous program

Teilprogramm, das von einem *Hintergrundauftrag* gestartet wird.

Asynchron-Vorgang (KDCS)
asynchronous service

Vorgang, der einen *Hintergrundauftrag* bearbeitet. Die Verarbeitung erfolgt entkoppelt vom Auftraggeber. Ein Asynchron-Vorgang kann aus einem oder mehreren Teilprogrammen/Transaktionen bestehen. Er wird über einen asynchronen *Transaktionscode* gestartet.

Auftrag
job

Anforderung eines *Services*, der von einer *UTM-Anwendung* zur Verfügung gestellt wird, durch Angabe eines Transaktionscodes. Siehe auch: *Ausgabeauftrag*, *Dialogauftrag*, *Hintergrundauftrag*, *Auftragskomplex*.

Auftraggeber-Vorgang
job-submitting service

Ein Auftraggeber-Vorgang ist ein *Vorgang*, der zur Bearbeitung eines Auftrags einen Service von einer anderen Server-Anwendung (*Auftragnehmer-Vorgang*) anfordert.

Auftragnehmer-Vorgang
job-receiving service

Ein Auftragnehmer-Vorgang ist ein *Vorgang*, der von einem *Auftraggeber-Vorgang* einer anderen Server-Anwendung gestartet wird.

Auftragskomplex
job complex

Auftragskomplexe dienen dazu, *Asynchronaufträgen Quittungsaufträge* zuzuordnen. Ein Asynchronauftrag innerhalb eines Auftragskomplexes wird *Basisauftrag* genannt.

Ausgabeauftrag
queued output job

Ausgabeaufträge sind *Asynchronaufträge*, die die Aufgabe haben, eine Nachricht, z.B. ein Dokument, an einen Drucker oder an ein Terminal auszugeben. Ausgabeziel kann aber auch eine andere Anwendung sein, die über die Transportsystem-Schnittstelle angeschlossen wurde.
Ausgabeaufträge werden ausschließlich von UTM-Systemfunktionen bearbeitet, d.h. für die Bearbeitung müssen keine Teilprogramme erstellt werden.

Authentisierung
authentication

siehe *Zugangskontrolle*.

Autorisierung
authorization
siehe *Zugriffskontrolle*.

Basisauftrag
basic job
Asynchronauftrag in einem *Auftragskomplex*.

Basisformat
basic format
Format, in das der Terminal-Benutzer alle Angaben eintragen kann, die notwendig sind, um einen Vorgang zu starten.

Benutzerausgang
user exit
Begriff ersetzt durch *Event-Exit.*

Benutzerkennung
user ID
Bezeichner für einen Benutzer, der in der *Konfiguration* der *UTM-Anwendung* festgelegt ist (optional mit Passwort zur *Zugangskontrolle*) und dem spezielle Zugriffsrechte (*Zugriffskontrolle*) zugeordnet sind. Ein Terminal-Benutzer oder ein *Client* muss bei der Anmeldung an die UTM-Anwendung diesen Bezeichner (und ggf. das zugeordnete Passwort) angeben. UTM-Anwendungen können auch ohne Benutzerkennungen generiert werden.

Benutzer-Protokolldatei
user log file
Datei oder Dateigeneration, in die der Benutzer mit dem LPUT-Aufruf Sätze variabler Länge schreibt. Jedem Satz werden die Daten aus dem KB-Kopf des *KDCS-Kommunikationsbereichs* vorangestellt. Die Benutzerprotokolldatei unterliegt der Transaktionssicherung von *open*UTM.

Berechtigungsprüfung
sign-on check
siehe *Zugangskontrolle*.

Beweissicherung (BS2000/OSD)
audit
Im Betrieb einer *UTM-Anwendung* werden zur Beweissicherung sicherheitsrelevante UTM-Ereignisse von *SAT* protokolliert.

Bildschirm-Wiederanlauf
screen restart

Wird ein *Dialog-Vorgang* unterbrochen, gibt *open*UTM beim *Vorgangswiederanlauf* die *Dialog-Nachricht* der letzten abgeschlossenen *Transaktion* erneut auf dem Bildschirm aus.

Browsen von Asynchron-Nachrichten
browsing asynchronous messages

Ein *Vorgang* liest nacheinander die *Asynchron-Nachrichten*, die sich in einer *Service-gesteuerten Queue* befinden. Die Nachrichten werden während des Lesens nicht gesperrt und verbleiben nach dem Lesen in der Queue. Dadurch ist gleichzeitiges Lesen durch unterschiedliche Vorgänge möglich.

Bypass-Betrieb (BS2000/OSD)
bypass mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Im Bypass-Betrieb wird eine an den Drucker gerichtete *Asynchron-Nachricht* an das Terminal gesendet und von diesem auf den Drucker umgeleitet, ohne auf dem Bildschirm angezeigt zu werden.

Cache-Speicher
cache

Pufferbereich zur Zwischenspeicherung von Anwenderdaten für alle Prozesse einer *UTM-Anwendung*. Der Cache-Speicher dient zur Optimierung der Zugriffe auf den *Pagepool*.

CCS-Name (BS2000/OSD)
CCS name

siehe *Coded-Character-Set-Name*.

Client
client

Clients einer *UTM-Anwendung* können sein:
- Terminals
- UPIC-Client-Programme
- Transportsystem-Anwendungen (z.B. DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als *Transportsystem-Anwendung* generiert sind)

Clients werden über LTERM-Partner an die UTM-Anwendung angeschlossen.

*open*UTM-Clients mit Trägersystem OpenCPIC werden wie *OSI-TP-Partner* behandelt.

Client-Seite einer Conversation
client side of a conversation
Begriff ersetzt durch *Initiator*.

Coded-Character-Set-Name (BS2000/OSD)
coded character set name
Bei Verwendung des Produkts *XHCS* (e**X**tended **H**ost **C**ode **S**upport) wird jeder verwendete Zeichensatz durch einen Coded-Character-Set-Namen (abgekürzt: „CCS-Name“ oder „CCSN“) eindeutig identifiziert.

Communication Resource Manager
communication resource manager
Communication Resource Manager (CRMs) kontrollieren in verteilten Systemen die Kommunikation zwischen den Anwendungsprogrammen. *open*UTM stellt CRMs für den internationalen Standard OSI TP, für den Industrie-Standard *LU6.1* und für das *open*UTM-eigene Protokoll UPIC zur Verfügung.

Contention Loser
contention loser
Bei der Kommunikation zwischen zwei Server-Anwendungen über LU6.1 ist der Contention Loser der Partner des *Contention Winner*.

Contention Winner
contention winner
Bei der Kommunikation zwischen zwei Server-Anwendungen über LU6.1 ist diejenige Anwendung der Contention Winner, die die Verbindung verwaltet.

Conversation
conversation
Bei CPI-C nennt man die Kommunikation zwischen zwei CPI-C-Anwendungsprogrammen Conversation. Die Kommunikationspartner einer Conversation werden *Initiator* und *Akzeptor* genannt.

Conversation-ID
conversation ID
Jeder *Conversation* wird von CPI-C lokal eine Conversation-ID zugeordnet, d.h. *Initiator* und *Akzeptor* haben jeweils eine eigene Conversation-ID. Mit der Conversation-ID wird jeder CPI-C-Aufruf innerhalb eines Programms eindeutig einer Conversation zugeordnet.

CPI-C

CPI-C (Common Programming Interface for Communication) ist eine von X/Open und dem CIW (**C**PI-C **I**mplementor's **W**orkshop) normierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen. Das in *open*UTM implementierte CPI-C genügt der CPI-C V2.0 CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. CPI-C in *open*UTM kann über die Protokolle OSI-TP, *LU6.1*, UPIC und mit *open*UTM-LU62 kommunizieren.

DES

DES (Data Encryption Standard) ist eine internationale Norm zur Verschlüsselung von Daten. Bei diesem Verfahren wird ein Schlüssel zum Ver- und Entschlüsseln verwendet. Wird das DES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen DES-Schlüssel.

Dialog-Auftrag

dialog job, interactive job

Auftrag, der einen *Dialog-Vorgang* startet. Der Auftrag kann von einem *Client* oder - bei *Server-Server-Kommunikation* - von einer anderen Anwendung (Client-Vorgang) erteilt werden.

Dialog-Conversation

dialog conversation

CPI-C-Conversation, bei der sowohl der *Initiator* als auch der *Akzeptor* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein Dialog-Transaktionscode generiert sein.

Dialog-Nachricht

dialog message

Nachricht, die eine Antwort erfordert oder selbst eine Antwort auf eine Anfrage ist. Dabei bilden Anfrage und Antwort einen *Dialogschritt*.

Dialogprogramm

dialog program

Teilprogramm, das einen *Dialogschritt* teilweise oder vollständig bearbeitet.

Dialogschritt

dialog step

Ein Dialogschritt beginnt mit dem Empfang einer *Dialog-Nachricht* durch die *UTM-Anwendung*. Er endet mit der Antwort der UTM-Anwendung.

Dialogterminalprozess (UNIX/Windows)
dialog terminal process

Ein Dialogterminalprozess verbindet ein UNIX/Windows-Terminal mit den *Work-prozessen* der *UTM-Anwendung*. Dialogterminalprozesse werden entweder vom Benutzer durch Eingabe von utmdtp oder über die LOGIN-Shell gestartet. Für jedes Terminal, das an eine UTM-Anwendung angeschlossen werden soll, ist ein eigener Dialogterminalprozess erforderlich.

Dialog-Vorgang
dialog service

Vorgang, der einen *Auftrag* im Dialog (zeitlich gekoppelt) mit dem Auftraggeber bearbeitet. Ein Dialog-Vorgang verarbeitet *Dialog-Nachrichten* vom Auftraggeber und erzeugt Dialog-Nachrichten für diesen. Ein Dialog-Vorgang besteht aus mindestens einer *Transaktion.* Ein Dialog-Vorgang umfasst in der Regel mindestens einen *Dialogschritt*. Ausnahme: Bei *Vorgangskettung* können auch mehrere Vorgänge einen Dialogschritt bilden.

Dienst
service

Programm, das im Hintergrund unabhängig von angemeldeten Benutzern oder Fenstern abläuft.

Dienstzugriffspunkt
service access point

Im *OSI-Referenzmodell* stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen *Selektor* identifiziert. Bei der Kommunikation bindet sich die *UTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Transaction Processing

X/Open-Architekturmodell für die transaktionsorientierte *verteilte Verarbeitung.*

Druckadministration
print administration

Funktionen zur *Drucksteuerung* und Administration von *Ausgabeaufträgen*, die an einen Drucker gerichtet sind.

Druckerbündel
printer pool

Mehrere Drucker, die demselben *LTERM-Partner* zugeordnet sind.

Druckergruppe (UNIX)
printer group
UNIX richtet für jeden Drucker standardmäßig eine Druckergruppe ein, die genau diesen Drucker enthält. Darüber hinaus lassen sich mehrere Drucker einer Druckergruppe, aber auch ein Drucker mehreren Druckergruppen zuordnen.

Druckerprozess (UNIX)
printer process
Prozess, der vom *Mainprozess* zur Ausgabe von *Asynchron-Nachrichten* an eine *Druckergruppe* eingerichtet wird. Er existiert, solange die Druckergruppe an die *UTM-Anwendung* angeschlossen ist. Pro angeschlossener Druckergruppe gibt es einen Druckerprozess.

Druckersteuerstation
printer control terminal
Begriff ersetzt durch *Druckersteuer-LTERM*

Druckersteuer-LTERM
printer control LTERM
Über ein Druckersteuer-LTERM kann sich ein *Client* oder ein Terminal-Benutzer an eine *UTM-Anwendung* anschließen. Von dem Client-Programm oder Terminal aus kann dann die *Administration* der Drucker erfolgen, die dem Druckersteuer-LTERM zugeordnet sind. Hierfür ist keine Administrationsberechtigung notwendig.

Drucksteuerung
print control
*open*UTM-Funktionen zur Steuerung von Druckausgaben.

Dynamische Konfigurierung
dynamic configuration
Änderung der *Konfiguration* durch die Administration. Im laufenden Betrieb der Anwendung können UTM-Objekte wie z.B. *Teilprogramme*, *Transaktionscodes*, *Clients*, *LU6.1-Verbindungen*, Drucker oder *Benutzerkennungen* in die Konfiguration aufgenommen, modifiziert oder teilweise auch gelöscht werden. Hierzu kann das Administrationsprogramm WinAdmin verwendet werden, oder es müssen eigene *Administrationsprogramme* erstellt werden, die die Funktionen der *Programmschnittstelle der Administration* nutzen.

Einschritt-Transaktion
single-step transaction
Transaktion, die genau einen *Dialogschritt* umfasst.

Einschritt-Vorgang
single-step service
Dialog-Vorgang, der genau einen *Dialogschritt* umfasst.

Ereignisgesteuerter Vorgang
event-driven service
Begriff ersetzt durch *Event-Service*.

Event-Exit
event exit
Routine des *Anwendungsprogramms*, das bei bestimmten Ereignissen (z.B. Start eines Prozesses, Ende eines Vorgangs) automatisch gestartet wird. Diese darf - im Gegensatz zu den *Event-Services* - keine KDCS-, CPI-C- und XATMI-Aufrufe enthalten.

Event-Funktion
event function
Oberbegriff für *Event-Exits* und *Event-Services*.

Event-Service
event service
Vorgang, der beim Auftreten bestimmter Ereignisse gestartet wird, z.B. bei bestimmten UTM-Meldungen. Die *Teilprogramme* ereignisgesteuerter Vorgänge müssen KDCS-Aufrufe enthalten.

Generierung
generation
Statische Konfigurierung einer *UTM-Anwendung* mit dem UTM-Tool KDCDEF und Erzeugen des Anwendungsprogramms.
Bei der Generierung entstehen die *KDCFILE* und die Source für das Tabellenmodul der Main-Routine *KDCROOT*.

Globaler Name
Symbolischer Name, den *TNS* zur Adressierung einer Anwendung benötigt. Unter dem globalen Namen werden die Adressen der lokalen als auch fernen Anwendungen in der TNS-Datenbasis abgelegt. Der globale Name der lokalen Anwendung ist der Anwendungsname (BCAMAPPL-Name, Name des OSI-TP-Zugriffspunkts). Der globale Name einer fernen Anwendung setzt sich zusammen aus Stationsname und Prozessorname.

Globaler Sekundärer Speicherbereich/GSSB
global secondary storage area
siehe *Sekundärspeicherbereich*.

Globale Transaktion
global transaction

Transaktion, die sich über mehr als eine Anwendung erstreckt.

Hardcopy-Betrieb
hardcopy mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Dabei wird eine Nachricht, die auf dem Bildschirm angezeigt wird, zusätzlich auf dem Drucker abgedruckt.

Heterogene Kopplung
heterogeneous link

Bei *Server-Server-Kommunikation:* Kopplung einer *UTM-Anwendung* mit einer Nicht-UTM-Anwendung, z.B. einer CICS- oder TUXEDO-Anwendung.

Hintergrundauftrag
background job

Hintergrund-Aufträge sind *Asynchron-Aufträge*, die an einen *Asynchron-Vorgang* der eigenen oder einer fernen Anwendung gerichtet sind. Hintergrund-Aufträge eignen sich besonders für zeitintensive oder zeitunkritische Verarbeitungen, deren Ergebnis keinen direkten Einfluss auf den aktuellen Dialog hat.

Homogene Kopplung
homogeneous link

Bei *Server-Server-Kommunikation*: Kopplung von *UTM-Anwendungen.* Dabei spielt es keine Rolle, ob die Anwendungen auf der gleichen oder auf unterschiedlichen Betriebssystem-Plattformen ablaufen.

Inbound-Conversation (CPI-C)
inbound conversation

siehe *Incoming-Conversation.*

Incoming-Conversation (CPI-C)
incoming conversation

Eine *Conversation*, bei der das lokale CPI-C-Programm *Akzeptor* ist, heißt Incoming-Conversation. In der X/Open-Specification wird für Incoming-Conversation auch das Synonym Inbound-Conversation verwendet.

Initiator (CPI-C)
initiator

Die Kommunikationspartner einer *Conversation* werden Initiator und *Akzeptor* genannt. Der Initiator baut die Conversation mit den CPI-C-Aufrufen Initialize_Conversation und Allocate auf.

Insert
insert

Feld in einem Meldungstext, in das *open*UTM aktuelle Werte einträgt.

Inverser KDCDEF
inverse KDCDEF

Funktion, die aus den Konfigurationsdaten der *KDCFILE*, die im laufenden Betrieb dynamisch angepasst wurde, Steueranweisungen für einen *KDCDEF*-Lauf erzeugt. Der inverse KDCDEF kann „offline“ unter KDCDEF oder „online“ über die *Programmschnittstelle zur Administration* gestartet werden.

Kaltstart
cold start

Starten einer *UTM-Anwendung* nach einer *normalen Beendigung* der Anwendung oder nach einer Neugenerierung (vgl. auch *Warmstart*).

KDCADM

Standard-Administrationsprogramm, das zusammen mit *open*UTM ausgeliefert wird. KDCADM stellt Administrationsfunktionen zur Verfügung, die über Transaktionscodes (*Administrationskommandos*) aufgerufen werden.

KDCDEF

UTM-Tool für die *Generierung* von *UTM-Anwendungen*. KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die UTM-Objekte *KDCFILE* und die ROOT-Tabellen-Source für die Main Routine *KDCROOT*.

KDCFILE

Eine oder mehrere Dateien, die für den Ablauf einer *UTM-Anwendung* notwendige Daten enthalten. Die KDCFILE wird mit dem UTM-Generierungstool *KDCDEF* erstellt. Die KDCFILE enthält unter anderem die *Konfiguration* der Anwendung.

KDCROOT

Main Routine eines *Anwendungsprogramms*, die das Bindeglied zwischen *Teilprogrammen* und UTM-Systemcode bildet. KDCROOT wird zusammen mit den *Teilprogrammen* zum *Anwendungsprogramm* gebunden.

KDCS-Parameterbereich
KDCS parameter area

siehe *Parameterbereich*.

KDCS-Programmschnittstelle
KDCS program interface

Universelle UTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen und *Message Queuing* Funktionen nutzen. Außerdem stellt KDCS Aufrufe zur *verteilten Verarbeitung* zur Verfügung.

Keycode
key code

Code, der in einer Anwendung eine bestimmte Zugriffsberechtigung oder eine bestimmte Rolle repräsentiert. Mehrere Keycodes werden zu einem *Keyset* zusammengefasst.

Keyset
key set

Zusammenfassung von einem oder mehrerer *Keycodes* unter einem bestimmten Namen. Ein Keyset definiert Berechtigungen im Rahmen des verwendeten Berechtigungskonzepts (Lock-/Keycode-Konzept oder *Access-List*-Konzept). Ein Keyset kann einer *Benutzerkennung*, einem *LTERM-Partner*, einem *Service* oder einer *TAC-Queue* zugeordnet werden.

Kommunikationsbereich/KB (KDCS)
communication area

Transaktionsgesicherter KDCS-*Primärspeicherbereich*, der vorgangsspezifische Daten enthält. Der Kommunikationsbereich besteht aus 3 Teilen:

- dem KB-Kopf mit allgemeinen Vorgangsdaten,
- dem KB-Rückgabebereich für Rückgaben nach KDCS-Aufrufen
- dem KB-Programmbereich zur Datenübergabe zwischen UTM-Teilprogrammen innerhalb eines *Vorgangs*.

Konfiguration
configuration

Summe aller Eigenschaften einer *UTM-Anwendung*. Die Konfiguration beschreibt:

- Anwendungs- und Betriebsparameter
- die Objekte der Anwendung und die Eigenschaften dieser Objekte. Objekte sind z.B. *Teilprogramme* und *Transaktionscodes*, Kommunikationspartner, Drucker, *Benutzerkennungen*
- definierte Zugriffsschutz- und Zugangsschutzmaßnahmen

Die Konfiguration einer UTM-Anwendung wird bei der Generierung festgelegt und kann per *Administration* dynamisch (während des Anwendungslaufs) geändert werden. Die Konfiguration ist in der *KDCFILE* abgelegt.

Konfigurierung
configuration

Festlegen der *Konfiguration* der UTM-Anwendung. Es wird unterschieden zwischen *statischer* und *dynamischer Konfigurierung*.

Logische Verbindung
virtual connection

Zuordnung zweier Kommunikationspartner.

Lockcode

Code, um einen LTERM-Partner oder einen Transaktionscode vor unberechtigtem Zugriff zu schützen. Damit ist ein Zugriff nur möglich, wenn das *Keyset* des Zugreifenden den passenden *Keycode* enthält (Lock-/Keycode-Konzept).

Lokaler Sekundärer Speicherbereich/LSSB
local secondary storage area

siehe *Sekundärspeicherbereich.*

LPAP-Partner
LPAP partner

Für die *verteilte Verarbeitung* über das *LU6.1*-Protokoll muss in der lokalen Anwendung für jede Partneranwendung ein LPAP-Partner konfiguriert werden. Der LPAP-Partner spiegelt in der lokalen Anwendung die Partneranwendung wider. Bei der Kommunikation wird die Partneranwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten LPAP-Partners angesprochen.

LTERM-Partner
LTERM partner

Um *Clients* oder Drucker an eine *UTM-Anwendung* anschließen zu können, müssen in der Anwendung LTERM-Partner konfiguriert werden. Ein Client oder Drucker kann nur angeschlossen werden, wenn ihm ein LTERM-Partner mit entsprechenden Eigenschaften zugeordnet ist. Diese Zuordnung wird i.A. bei der Konfigurierung festgelegt, sie kann aber auch dynamisch über Terminal-Pools erfolgen.

LTERM-Pool
LTERM pool

Statt für jeden *Client* eine LTERM- und eine PTERM-Anweisung anzugeben, kann mit der Anweisung TPOOL ein Pool von LTERM-Partnern definiert werden. Schließt sich ein Client über einen LTERM-Pool an, wird ihm dynamisch ein LTERM-Partner aus dem Pool zugeordnet.

LU6.1

Geräteunabhängiges Datenaustauschprotokoll (Industrie-Standard) für die transaktionsgesicherte *Server-Server-Kommunikation*.

Mainprozess (UNIX/Windows)

main process

Prozess, der die *UTM-Anwendung* startet. Er startet außerdem die *Workprozesse*, die *Druckerprozesse, Netzprozesse* und den *Timerprozess* und überwacht und steuert die *UTM-Anwendung*.

Main Routine KDCROOT

main routine KDCROOT

siehe *KDCROOT*.

Meldung

UTM message

Meldungen werden von dem Transaktionsmonitor *open*UTM oder von UTM-Tools (wie z.B. *KDCDEF*) an *Meldungsziele* ausgegeben. Eine Meldung besteht aus einer Meldungsnummer und dem Meldungstext, der ggf. *Inserts* mit aktuellen Werten enthält. Je nach Meldungsziel werden entweder die gesamte Meldung oder nur Teile der Meldung (z.B. nur die Inserts) ausgegeben.

Meldungsdefinitionsdatei

UTM message definition file

Die Meldungsdefinitionsdatei wird mit *open*UTM ausgeliefert und enthält standardmäßig die UTM-Meldungstexte in deutscher und englischer Sprache und die Definitionen der Meldungseigenschaften. Aufbauend auf diese Datei kann der Anwender auch eigene, individuelle Meldungsmodule erzeugen.

Meldungsziel

UTM message destination

Ausgabemedium für eine *Meldung*. Mögliche Meldungsziele einer Meldung sind z.B. Terminals, *TS-Anwendungen*, der *Event-Service* MSGTAC, die *System-Protokolldatei* SYSLOG oder *TAC-Queues, Asynchron-TACs* und *USER-Queues*.

Mehrschritt-Transaktion

multi-step transaction

Transaktion, die aus mehr als einem *Verarbeitungsschritt* besteht.

Mehrschritt-Vorgang (KDCS)

multi-step service

Vorgang, der in mehreren *Dialogschritten* ausgeführt wird.

Message Queuing
message queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete *Message Queues* ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen, die Übermittlung der Nachricht wird garantiert, unabhängig davon, ob gerade eine Netzverbindung besteht oder nicht. Bei *open*UTM gibt es *UTM-gesteuerte Queues* und *Service-gesteuerte Queues*.

Message Queue
message queue

Warteschlange, in der bestimmte Nachrichten transaktionsgesichert bis zur Weiterverarbeitung eingereiht werden. Je nachdem, wer die Weiterverarbeitung kontrolliert, unterscheidet man *Service-gesteuerte Queues* und *UTM-gesteuerte Queues*.

MSGTAC
MSGTAC

Spezieller Event-Service, der Meldungen mit dem Meldungsziel MSGTAC per Programm verarbeitet. MSGTAC ist ein Asynchron-Vorgang und wird vom Betreiber der Anwendung erstellt.

Multiplexanschluss (BS2000/OSD)
multiplex connection

Spezielle Möglichkeit, Terminals an eine *UTM-Anwendung* anzuschließen. Ein Multiplexanschluss ermöglicht es, dass sich mehrere Terminals eine *Transportverbindung* teilen.

Nachrichtenbereich/NB (KDCS)
KDCS message area

Bei KDCS-Aufrufen: Puffer-Bereich, in dem Nachrichten oder Daten für *open*UTM oder für das *Teilprogramm* bereitgestellt werden.

Nachrichtenverteiler (BS2000/OSD)
message router

Einrichtung in einem zentralen Rechner oder Kommunikationsrechner zur Verteilung von Eingabenachrichten an unterschiedliche *UTM-Anwendungen*, die auf unterschiedlichen Rechnern liegen können. Der Nachrichtenverteiler ermöglicht außerdem, mit *Multiplexanschlüssen* zu arbeiten.

Netzprozess (UNIX/Windows)
net process

Prozess einer *UTM-Anwendung* zur Netzanbindung.

Netzwerk-Selektor
network selector

Der Netzwerk-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Vermittlungsschicht des *OSI-Referenzmodells*.

Normale Beendigung einer UTM-Anwendung
normal termination of a UTM application

Kontrollierte Beendigung einer *UTM-Anwendung*; das bedeutet u.a., dass die Verwaltungsdaten auf der *KDCFILE* aktualisiert werden. Eine normale Beendigung veranlasst der *Administrator* (z.B. mit KDCSHUT N). Den Start nach einer normalen Beendigung führt *open*UTM als *Kaltstart* durch.

Object Identifier
object identifier

Ein Object Identifier ist ein global (d.h. weltweit) eindeutiger Bezeichner für Objekte im OSI-Umfeld. Ein Object Identifier besteht aus einer Folge von ganzen Zahlen, die einen Pfad in einer Baumstruktur repräsentiert.

Offener Terminalpool
open terminal pool

Terminalpool, der nicht auf *Clients* eines Rechners oder eines bestimmten Typs beschränkt ist. An diesen Terminalpool können sich alle Clients anschließen, für die kein rechner- oder typspezifischer Terminalpool generiert ist.

***open*UTM-Anwendung**
*open*UTM application

siehe *UTM-Anwendung*.

***open*UTM-D**

*open*UTM-D (*open*UTM-Distributed) ist eine *open*UTM-Komponente, die *verteilte Verarbeitung* ermöglicht. *open*UTM-D ist integraler Bestandteil von *open*UTM.

OSI-LPAP-Partner
OSI-LPAP partner

OSI-LPAP-Partner sind die bei *open*UTM generierten Adressen der *OSI-TP-Partner*. Für die *verteilte Verarbeitung* über das Protokoll *OSI TP* muss in der lokalen Anwendung für jede Partneranwendung ein OSI-LPAP-Partner konfiguriert werden. Der OSI-LPAP-Partner spiegelt in der lokalen Anwendung die Partneranwendung wider. Bei der Kommunikation wird die Partneranwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell
OSI reference model

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Von der ISO definiertes Kommunikationsprotokoll für die verteilte Transaktionsverarbeitung. OSI TP steht für Open System Interconnection Transaction Processing.

OSI-TP-Partner
OSI-TP partner

Partner der UTM-Anwendung, der mit der UTM-Anwendung über das OSI-TP-Protokoll kommuniziert.
Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über OSI TP kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS), die über *open*UTM-LU62 angeschlossen ist
- eine Anwendung des Trägersystems OpenCPIC des *open*UTM-Client
- Anwendungen anderer TP-Monitore, die OSI TP unterstützen

Outbound-Conversation (CPI-C)
outbound conversation

siehe *Outgoing-Conversation*.

Outgoing-Conversation (CPI-C)
outgoing conversation

Eine Conversation, bei der das lokale CPI-C-Programm der *Initiator* ist, heißt Outgoing-Conversation. In der X/Open-Specification wird für Outgoing-Conversation auch das Synonym Outbound-Conversation verwendet.

Pagepool
page pool

Teil der *KDCFILE*, in dem Anwenderdaten gespeichert werden, z.B. *Dialog-Nachrichten*, Nachrichten an *Message Queues*, *Sekundärspeicherbereiche*.

Parameterbereich
parameter area
: Datenstruktur, in der ein *Teilprogramm* bei einem UTM-Aufruf die für diesen Aufruf notwendigen Operanden an *open*UTM übergibt.

Postselection (BS2000/OSD)
postselection
: Auswahl der protokollierten UTM-Ereignisse aus der SAT-Protokolldatei, die ausgewertet werden sollen. Die Auswahl erfolgt mit Hilfe des Tools SATUT.

Prädialog (BS2000/OSD)
predialog
: Aufforderung des Terminal-Benutzers an das Datenkommunikationssystem zum Aufbau einer *logischen Verbindung* zur *Anwendung*. Der Prädialog entfällt, wenn die logische Verbindung auf Initiative der Anwendung aufgebaut wird.

Preselection (BS2000/OSD)
preselection
: Festlegung der für die *SAT-Beweissicherung* zu protokollierenden UTM-Ereignisse. Die Preselection erfolgt durch die UTM-SAT-Administration. Man unterscheidet ereignisspezifische, benutzerspezifische und auftrags-(TAC-)spezifische Preselection.

Presentation-Selektor
presentation selector
: Der Presentation-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Darstellungsschicht des *OSI-Referenzmodells*.

Primärspeicherbereich
primary storage area
: Bereich im Arbeitsspeicher, auf den das *KDCS-Teilprogramm* direkt zugreifen kann, z.B. *Standard Primärer Arbeitsbereich*, *Kommunikationsbereich*.

Printerprozess (UNIX)
printer process
: siehe *Druckerprozess*.

Programmschnittstelle zur Administration
program interface for administration
: UTM-Programmschnittstelle, mit deren Hilfe der Anwender eigene *Administrationsprogramme* erstellen kann. Die Programmschnittstelle zur Administration bietet u.a. Funktionen zur *dynamischen Konfigurierung*, zur Modifikation von Eigenschaften und Anwendungsparametern und zur Abfrage von Informationen zur *Konfiguration* und zur aktuellen Auslastung der Anwendung.

Prozess
process

In den *open*UTM-Handbüchern wird der Begriff „Prozess“ als Oberbegriff für Prozess (UNIX/Windows) und Task (BS2000/OSD) verwendet.

Queue
queue

siehe *Message Queue*

QuickStart Kit

Beispielanwendung, die mit *open*UTM (Windows) ausgeliefert wird.

Quittungsauftrag
confirmation job

Bestandteil eines *Auftragskomplexes*, worin der Quittungsauftrag dem *Basisauftrag* zugeordnet ist. Es gibt positive und negative Quittungsaufträge. Bei positivem Ergebnis des *Basisauftrags* wird der positive Quittungsauftrag wirksam, sonst der negative.

Redelivery
redelivery

Erneutes Zustellen einer *Asynchron-Nachricht*, nachdem diese nicht ordnungsgemäß verarbeitet werden konnte, z.B. weil die *Transaktion* zurückgesetzt oder der *Asynchron-Vorgang* abnormal beendet wurde. Die Nachricht wird wieder in die Message Queue eingereiht und lässt sich damit erneut lesen und/oder verarbeiten.

Reentrant-fähiges Programm
reentrant program

Programm, dessen Code durch die Ausführung nicht verändert wird.
Im BS2000/OSD ist dies Voraussetzung dafür, *Shared Code* zu nutzen.

Request
request

Anforderung einer *Service-Funktion* durch einen *Client* oder einen anderen Server.

Requestor
requestor

In XATMI steht der Begriff Requestor für eine Anwendung, die einen Service aufruft.

Resource Manager
resource manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme. *open*UTM stellt aber auch selbst Resource Manager zur Verfügung, z.B. für den Zugriff auf *Message Queues*, lokale Speicherbereiche und Logging-Dateien. Anwendungsprogramme greifen auf RMs über RM-spezifische Schnittstellen zu. Für Datenbank-Systeme ist dies meist SQL, für die *open*UTM-RMs die Schnittstelle KDCS.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

RSA

Abkürzung für die Erfinder dieses Verschlüsselungsverfahrens Rivest, Shamir und Adleman. Bei diesem Verfahren wird ein Schlüsselpaar verwendet, das aus einem öffentlichen und einem privaten Schlüssel besteht. Eine Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und kann nur mit dem privaten Schlüssel entschlüsselt werden. Das RSA-Schlüsselpaar wird von der UTM-Anwendung erzeugt.

SAT-Beweissicherung (BS2000/OSD)
SAT audit

Beweissicherung durch die Komponente SAT (Security Audit Trail) des BS2000-Softwareproduktes SECOS.

Sekundärspeicherbereich
secondary storage area

Transaktionsgesicherter Speicherbereich auf der *KDCFILE*, auf den das KDCS-*Teilprogramm* mit speziellen Aufrufen zugreifen kann. Lokale Sekundärspeicherbereiche (LSSB) sind einem *Vorgang* zugeordnet, auf globale Sekundärspeicherbereiche (GSSB) kann von allen Vorgängen einer *UTM-Anwendung* zugegriffen werden. Weitere Sekundärspeicherbereiche sind der *Terminalspezifische Langzeitspeicher (TLS)* und der *Userspezifische Langzeitspeicher (ULS)*.

Selektor
selector

Ein Selektor identifiziert im lokalen System einen *Zugriffspunkt* auf die Dienste einer Schicht des *OSI-Referenzmodells*. Jeder Selektor ist Bestandteil der Adresse des Zugriffspunktes.

Semaphor (UNIX/Windows)
semaphore
UNIX- und Windows-Betriebsmittel, das zur Steuerung und Synchronisation von Prozessen dient.

Server
server
Ein Server ist eine *Anwendung*, die *Services* zur Verfügung stellt. Oft bezeichnet man auch den Rechner, auf dem Server-Anwendungen laufen, als Server.

Server-Seite einer Conversation (CPI-C)
server side of a conversation
Begriff ersetzt durch *Akzeptor*.

Server-Server-Kommunikation
server-server communication
siehe *verteilte Verarbeitung*.

Service Access Point
siehe *Dienstzugriffspunkt.*

Service
service
Services sind Dienstleistungen, die von *Servern* erbracht werden. Diese Services können von *Clients* oder anderen Servern aufgerufen werden.

Service-gesteuerte Queue
service controlled queue
Message Queue, bei der der Abruf und die Weiterverarbeitung der Nachrichten durch *Services* gesteuert werden. Ein Service muss zum Lesen der Nachricht explizit einen KDCS-Aufruf (DGET) absetzen.
Service-gesteuerte Queues gibt es bei *open*UTM in den Varianten *USER-Queue*, *TAC-Queue* und *Temporäre Queue*.

Service Routine
service routine
siehe *Teilprogramm*.

Session
session
Kommunikationsbeziehung zweier adressierbarer Einheiten im Netz über das SNA-Protokoll *LU6.1*.

Session-Selektor
session selector

Der Session-Selektor identifiziert im lokalen System einen *Zugriffspunkt* zu den Diensten der Kommunikationssteuerschicht (Session-Layer) des *OSI-Referenzmodells*.

Shared Code (BS2000/OSD)
shared code

Code, der von mehreren Prozessen gleichzeitig benutzt werden kann.

Shared Memory
shared memory

Virtueller Speicherbereich, auf den mehrere Prozesse gleichzeitig zugreifen können.

Shared Objects (UNIX/Windows)
shared objects

Teile des *Anwendungsprogramms* können als Shared Objects erzeugt werden. Diese werden dynamisch zur Anwendung dazugebunden und können im laufenden Betrieb ausgetauscht werden. Shared Objects werden mit der KDCDEF-Anweisung SHARED-OBJECT definiert.

Sicherungspunkt
synchronization point, consistency point

Ende einer *Transaktion*. Zu diesem Zeitpunkt werden alle in der Transaktion vorgenommenen Änderungen der *Anwendungsinformation* gegen Systemausfall gesichert und für andere sichtbar gemacht. Während der Transaktion gesetzte Sperren werden wieder aufgehoben.

Socket-Verbindung
socket connection

Transportsystem-Verbindung, die die Socket-Schnittstelle verwendet. Die Socket-Schnittstelle ist eine Standard-Programmschnittstelle für die Kommunikation über TCP/IP.

Standard Primärer Arbeitsbereich/SPAB (KDCS)
standard primary working area

Bereich im Arbeitsspeicher, der jedem KDCS-*Teilprogramm* zur Verfügung steht. Sein Inhalt ist zu Beginn des Teilprogrammlaufs undefiniert oder mit einem Füllzeichen vorbelegt.

Startformat
start format

Format, das *open*UTM am Terminal ausgibt, wenn sich ein Benutzer erfolgreich bei der *UTM-Anwendung* angemeldet hat (ausgenommen nach *Vorgangswiederanlauf* und beim *Anmeldevorgang*).

statische Konfigurierung
static configuration

Festlegen der *Konfiguration* bei der Generierung mit Hilfe des UTM-Tools *KDCDEF*.

SYSLOG-Datei
SYSLOG file

siehe *System-Protokolldatei*.

System-Protokolldatei
system log file

Datei oder Dateigeneration, in die *open*UTM während des Laufs einer *UTM-Anwendung* alle UTM-Meldungen protokolliert, für die das *Meldungsziel* SYSLOG definiert ist.

TAC
TAC

siehe *Transaktionscode*.

TAC-Queue
TAC queue

Message Queue, die explizit per KDCDEF-Anweisung generiert wird. Eine TAC-Queue ist eine *Service-gesteuerte Queue* und kann unter dem generierten Namen von jedem Service aus angesprochen werden.

Teilprogramm
program unit

UTM-*Services* werden durch ein oder mehrere Teilprogramme realisiert. Die Teilprogramme sind Bestandteile des *Anwendungsprogramms*. Sie enthalten UTM-Funktionsaufrufe und sind über *Transaktionscodes* ansprechbar. Einem Teilprogramm können mehrere Transaktionscodes zugeordnet werden.

Temporäre Queue
temporary queue

Message Queue, die dynamisch per Programm erzeugt wird und auch wieder per Programm gelöscht werden kann, vgl. *Service-gesteuerte Queue*.

Terminalspezifischer Langzeitspeicher/TLS (KDCS)
terminal-specific long-term storage

Sekundärspeicher, der einem *LTERM-, LPAP-* oder *OSI-LPAP-Partner* zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

Timerprozess (UNIX/Windows)
timer process

Prozess, der Aufträge zur Zeitüberwachung von *Workprozessen* entgegennimmt, sie in ein Auftragsbuch einordnet und nach einer im Auftragsbuch festgelegten Zeit den Workprozessen zur Bearbeitung wieder zustellt.

TNS (UNIX/Windows)

Abkürzung für den Transport Name Service, der einem Anwendungsnamen einen Transport-Selektor und das Transportsystem zuordnet, über das die Anwendung erreichbar ist.

Transaktion
transaction

Verarbeitungsabschnitt innerhalb eines *Services*, der die *ACID-Eigenschaften* aufweist. Von den in einer Transaktion beabsichtigten Änderungen der *Anwendungsinformation* werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt (Alles-oder-Nichts Regel). Das Transaktionsende bildet einen *Sicherungspunkt*.

Transaktionscode/TAC
transaction code

Name, über den ein *Teilprogramm* aufgerufen werden kann. Der Transaktionscode wird dem Teilprogramm bei der *statischen* oder *dynamischen Konfigurierung* zugeordnet. Einem Teilprogramm können auch mehrere Transaktionscodes zugeordnet werden.

Transaktionsrate
transaction rate

Anzahl der erfolgreich beendeten *Transaktionen* pro Zeiteinheit.

Transfer-Syntax
transfer syntax

Bei *OSI TP* werden die Daten zur Übertragung zwischen zwei Rechnersystemen von der lokalen Darstellung in die Transfer-Syntax umgewandelt. Die Transfer-Syntax beschreibt die Daten in einem neutralen Format, das von allen beteiligten Partnern verstanden wird. Jeder Transfer-Syntax muss ein *Object Identifier* zugeordnet sein.

Transport-Selektor
transport selector

Der Transport-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Transportschicht des *OSI-Referenzmodells*.

Transportsystem-Anwendung
transport system application

Anwendung, die direkt auf der Transportsystem-Schnittstelle wie z.B. CMX oder Socket aufsetzt. Für den Anschluss von Transportsystem-Anwendungen muss bei der Konfigurierung als Partnertyp APPLI oder SOCKET angegeben werden. Eine Transportsystem-Anwendung kann nicht in eine globale Transaktionssicherung eingebunden werden.

TS-Anwendung
TS application

siehe *Transportsystem-Anwendung*.

Typisierter Puffer (XATMI)
typed buffer

Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten den Partnern implizit bekannt.

UPIC

Trägersystem für *open*UTM-Clients. UPIC steht für Universal Programming Interface for Communication.

UPIC-Client

Bezeichnung für *open*UTM-Clients mit Trägersystem UPIC

USER-Queue
USER queue

Message Queue, die *open*UTM jeder Benutzerkennung zur Verfügung stellt. Eine USER-Queue zählt zu den *Service-gesteuerten Queues* und ist immer der jeweiligen Benutzerkennung zugeordnet. Der Zugriff von fremden UTM-Benutzern auf die eigene USER-Queue kann eingeschränkt werden.

Userspezifischer Langzeitspeicher/ULS
user-specific long-term storage

Sekundärspeicher, der einer *Benutzerkennung,* einer *Session* oder einer *Association* zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

USLOG-Datei
USLOG file

siehe *Benutzer-Protokolldatei.*

UTM-Anwendung
UTM application

Eine UTM-Anwendung stellt *Services* zur Verfügung, die Aufträge von *Clients* oder anderen *Server*-Anwendungen bearbeiten. *open*UTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine UTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Server-Einheit bildet.

UTM-D

siehe *openUTM-D.*

UTM-Datenstation
UTM terminal

Begriff ersetzt durch *LTERM-Partner*.

UTM-F

UTM-Anwendungen können als UTM-F-Anwendungen (UTM-Fast) generiert werden. Bei UTM-F wird zugunsten der Performance auf Platteneingaben/-ausgaben verzichtet, mit denen bei *UTM-S* die Sicherung von Benutzer- und Transaktionsdaten durchgeführt wird. Gesichert werden lediglich Änderungen der Verwaltungsdaten.

UTM-gesteuerte Queues
UTM controlled queue

Message Queues, bei denen der Abruf und die Weiterverarbeitung der Nachrichten vollständig durch *open*UTM gesteuert werden. Siehe auch *Asynchron-Auftrag, Hintergrundauftrag* und *Asynchron-Nachricht*.

UTM-S

Bei UTM-S-Anwendungen sichert *open*UTM neben den Verwaltungsdaten auch alle Benutzerdaten über ein Anwendungsende und einen Systemausfall hinaus. Außerdem garantiert UTM-S bei allen Störungen die Sicherheit und Konsistenz der Anwendungsdaten. Im Standardfall werden UTM-Anwendungen als UTM-S-Anwendungen (UTM-Secure) generiert.

UTM-SAT-Administration (BS2000/OSD)
UTM SAT administration

Durch die UTM-SAT-Administration wird gesteuert, welche sicherheitsrelevanten UTM-Ereignisse, die im Betrieb der *UTM-Anwendung* auftreten, von *SAT* protokolliert werden sollen. Für die UTM-SAT-Administration wird eine besondere Berechtigung benötigt.

UTM-Seite
UTM page

Ist eine Speichereinheit, die entweder 2K oder 4K umfasst. Die Größe wird bei der Generierung der *UTM-Anwendung* festgelegt. *Pagepool* und Wiederanlauf-Bereich der KDCFILE werden in Einheiten der Größe einer UTM-Seite unterteilt.

Verarbeitungsschritt
processing step

Ein Verarbeitungsschritt beginnt mit dem Empfangen einer *Dialog-Nachricht*, die von einem *Client* oder einer anderen Server-Anwendung an die *UTM-Anwendung* gesendet wird. Der Verarbeitungsschritt endet entweder mit dem Senden einer Antwort und beendet damit auch den *Dialogschritt* oder er endet mit dem Senden einer Dialog-Nachricht an einen Dritten.

Verschlüsselungsstufe
encryption level

Die Verschlüsselungsstufe legt fest, ob und inwieweit ein Client Nachrichten und Passwort verschlüsseln muss.

Verteilte Transaktion
distributed transaction

siehe *globale Transaktion*.

Verteilte Transaktionsverarbeitung
Distributed Transaction Processing

Verteilte Verarbeitung mit *globalen Transaktionen*.

Verteilte Verarbeitung
distributed processing

Bearbeitung von *Dialogaufträgen* durch mehrere Anwendungen oder Übermittlung von *Hintergrundaufträgen* an eine andere Anwendung. Für die verteilte Verarbeitung werden die höheren Kommunikationsprotokolle *LU6.1* und *OSI TP* verwendet. Über *open*UTM-LU62 ist verteilte Verarbeitung auch mit LU6.2 Partnern möglich. Man unterscheidet verteilte Verarbeitung mit *globalen Transaktionen* (anwendungsübergreifende Transaktionssicherung) und verteilte Verarbeitung ohne globale Transaktionen (nur lokale Transaktionssicherung). Die verteilte Verarbeitung wird auch Server-Server-Kommunikation genannt.

Vorgang (KDCS)
service

Ein Vorgang dient zur Bearbeitung eines *Auftrags* in einer *UTM-Anwendung*. Er setzt sich aus einer oder mehreren *Transaktionen* zusammen. Die erste Transaktion wird über den *Vorgangs-TAC* aufgerufen. Es gibt *Dialog-Vorgänge* und *Asynchron-Vorgänge*. *open*UTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung. Anstelle des Begriffs Vorgang wird häufig auch der allgemeinere Begriff *Service* gebraucht.

Vorgangskellerung (KDCS)
service stacking

Ein Terminal-Benutzer kann einen laufenden *Dialog-Vorgang* unterbrechen und einen neuen Dialog-Vorgang einschieben. Nach Beendigung des eingeschobenen *Vorgangs* wird der unterbrochene Vorgang fortgesetzt.

Vorgangskettung (KDCS)
service chaining

Bei Vorgangskettung wird nach Beendigung eines *Dialog-Vorgangs* ohne Angabe einer *Dialog-Nachricht* ein Folgevorgang gestartet.

Vorgangs-TAC (KDCS)
service TAC

Transaktionscode, mit dem ein *Vorgang* gestartet wird.

Vorgangswiederanlauf (KDCS)
service restart

Nach einem Vorgangsabbruch, z.B. infolge Abmeldens des Terminal-Benutzers oder Beendigung der *UTM-Anwendung,* führt *open*UTM einen Vorgangswiederanlauf durch. Ein *Asynchron-Vorgang* wird neu gestartet oder beim zuletzt erreichten *Sicherungspunkt* fortgesetzt, ein *Dialog-Vorgang* wird beim zuletzt erreichten Sicherungspunkt fortgesetzt. Für den Terminal-Benutzer wird der Vorgangswiederanlauf eines Dialog-Vorgangs als *Bildschirm-Wiederanlauf* sichtbar.

Warmstart
warm start

Starten einer *UTM-Anwendung* nach einer *abnormalen Beendigung*. Beim Warmstart wird die *KDCFILE* wieder in einen konsistenten Zustand gebracht.

Wiederanlauf
restart

siehe *Anwendungswiederanlauf*,
siehe *Bildschirm-Wiederanlauf*,
siehe *Vorgangswiederanlauf*.

Workprozess (UNIX/Windows)
work process

Prozess, in dem die *Services* der *UTM-Anwendung* ablaufen.

XATMI

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen.
Das in *open*UTM implementierte XATMI genügt der XATMI CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. XATMI in *open*UTM kann über die Protokolle OSI-TP, *LU6.1* und UPIC kommunizieren.

XHCS (BS2000/OSD)

XHCS (Extended Host Code Support) ist ein BS2000/OSD-Softwareprodukt für die Unterstützung internationaler Zeichensätze.

XML

XML (eXtensible Markup Language) ist eine vom W3C (WWW-Konsortium) genormte Metasprache, in der Austauschformate für Daten und zugehörige Informationen definiert werden können.

Zeitgesteuerter Auftrag
time-driven job

Auftrag, der von *open*UTM bis zu einem definierten Zeitpunkt in einer *Message Queue* zwischengespeichert und dann an den Empfänger weitergeleitet wird. Empfänger kann sein: ein *Asynchron-Vorgang* der selben Anwendung, eine *TAC-Queue*, eine Partneranwendung, ein Terminal oder ein Drucker.
Zeitgesteuerte Aufträge können nur von KDCS-*Teilprogrammen* erteilt werden.

Zugangskontrolle
system access control

Prüfung durch *open*UTM, ob eine bestimmte *Benutzerkennung* berechtigt ist, mit der *UTM-Anwendung* zu arbeiten. Die Berechtigungsprüfung entfällt, wenn die UTM-Anwendung ohne Benutzerkennungen generiert wurde.

Zugriffskontrolle
data access control

Prüfung durch *open*UTM, ob der Kommunikationspartner berechtigt ist, auf ein bestimmtes Objekt der Anwendung zuzugreifen. Die Zugriffsrechte werden als Bestandteil der Konfiguration festgelegt.

Zugriffspunkt
access point

siehe *Dienstzugriffspunkt.*

Abkürzungen

ACSE	Association Control Service Element
AEQ	Application Entity Qualifier
AES	Advanced Encryption Standard
AET	Application Entity Title
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
BCAM	Basic Communication Access Method
BER	Basic Encoding Rules
BLS	Binder-Lader-Starter (BS2000/OSD)
BSI	Bundesamt für Sicherheit in der Informationstechnik
CCP	Communication Control Program
CCR	Commitment, Concurrency and Recovery
CCS	Codierter Zeichensatz (Coded Character Set)
CCSN	Name des codierten Zeichensatzes (Coded Character Set Name)
CICS	Customer Information Control System
CID	Control Identification
CIW	CPI-C Implementor’s Workshop
CMX	Communication Manager in UNIX
COM	Component Object Model
CPI-C	Common Programming Interface for Communication
CRM	Communication Resource Manager
CRTE	Common Runtime Environment (BS2000/OSD)
DB	Datenbank
DC	Data Communication

DCAM	Data Communication Access Method
DCOM	Distributed Component Object Mode
DES	Data Encryption Standard
DMS	Data Management System
DNS	Domain Name Service
DSS	Datensichtstation (=Terminal)
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DVS	Datenverwaltungssystem
EBCDIC	Binärcode für die stellenweise Verschlüsselung von Dezimalziffern (Extended Binary-Coded Decimal Interchange Code)
EJB	Enterprise JavaBeans*TM*
FGG	File Generation Group
FHS	Format Handling System
FORMANT	Format-Manager für Terminals
FT	File Transfer
GSSB	Globaler Sekundärer Speicherbereich
HLL	High-Level Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IFG	Interaktiver Format-Generator
ILCS	Inter Language Communication Services (BS2000/OSD)
IMS	Information Management System (IBM)
IPC	Inter-Process-Communication
IRV	Internationale Referenzversion
ISO	International Organization for Standardization
JCA	Java Connector Architecture
KAA	KDCS Application Area
KB	Kommunikationsbereich
KBPROG	KB-Programmbereich
KDCS	Kompatible Datenkommunikationsschnittstelle
KTA	KDCS Task Area
LAN	Local Area Network

LCF	Local Configuration File
LLM	Link and Load Module (BS2000/OSD)
LSSB	Lokaler Sekundärer Speicherbereich
LU	Logical Unit
MIGRAT	Migrationsprogramm
MQ	Message Queuing
MSCF	Multiple System Control Facility (BS2000/OSD)
MSF	Mehrfachsteuerung für Fernanschluss
MSN	Mehrfachsteuerung für Nahbereich
NEA	Netzwerkarchitektur bei TRANSDATA-Systemen
NB	Nachrichtenbereich
NLS	Unterstützung der Landessprache (Native Language Support)
OCX	OLE Control Extension
OML	Object Modul Library
OLTP	Online Transaction Processing
OSI	Open System Interconnection
OSI-TP	Open System Interconnection Transaction Processing
OSS	OSI Session Service
PCMX	Portable Communication Manager
PDN	Programmsystem für Datenfernverarbeitung und Netzsteuerung
PID	Prozess-Identifikation
PIN	Persönliche Identifikationsnummer
PLU	Primary Logical Unit
RAV	Rechenzentrums-Abrechnungs-Verfahren
RDF	Resource Definition File
RM	Resource Manager
RSA	Encryption-Algorithmus nach Rivest, Shamir, Adleman
RSO	Remote SPOOL Output (BS2000/OSD)
RTS	Runtime System (Laufzeitsystem)
SAT	Security Audit Trail (BS2000/OSD)
SECOS	Security Control System
SGML	Standard Generalized Markup Language

SLU	Secondary Logical Unit
SM2	Software Monitor 2 (BS2000/OSD)
SNA	Systems Network Architecture
SPAB	Standard Primärer Arbeitsbereich
SQL	Structured Query Language
SSB	Sekundärer Speicherbereich
SSO	Single-Sign-On
TAC	Transaktionscode
TCP/IP	Transport Control Protocol / Internet Protocol
TIAM	Terminal Interactive Access Method
TLS	Terminalspezifischer Langzeitspeicher
TM	Transaction Manager
TNS	Transport Name Service
TNSX	Transport Name Service in UNIX / Windows
TP	Transaction Processing (Transaktions-Betrieb)
TPR	Task privileged (privilegierter Funktionszustand des BS2000/OSD)
TPSU	Transaction Protocol Service User
TSN	Task Sequence Number
TU	Task user (nicht privilegierter Funktionszustand des BS2000/OSD)
TX	Transaction Demarcation (X/Open)
UDS	Universelles Datenbanksystem
UDT	Unstructured Data Transfer
ULS	Userspezifischer Langzeitspeicher
UPIC	Universal Programming Interface for Communication
USP	UTM-Socket-Protokoll
UTM	Universeller Transaktionsmonitor
UTM-D	UTM-Funktionen für verteilte Verarbeitung („Distributed“)
UTM-F	Schnelle UTM-Variante („Fast“)
UTM-S	UTM-Sicherheitsvariante
VGID	Vorgangsidentifikation
VTSU	Virtual Terminal Support
VTV	Verteilte Transaktionsverarbeitung
VV	Verteilte Verarbeitung

WAN	Wide Area Network
XA	X/Open Access Interface (Schnittstelle von X/Open zum Zugriff auf Resource Manager)
XAP	X/OPEN ACSE/Presentation programming interface
XAP-TP	X/OPEN ACSE/Presentation programming interface Transaction Processing extension
XATMI	X/Open Application Transaction Manager Interface
XHCS	eXtended Host Code Support
XML	eXtensible Markup Language

Literatur

Zu *open*UTM gibt es auch eine CD-ROM. Diese enthält PDF-Dateien von allen *open*UTM-Handbüchern.
Die Handbücher sind online unter *http://manuals.fujitsu-siemens.com* zu finden oder in gedruckter Form gegen gesondertes Entgelt unter *http://FSC-manualshop.com* zu bestellen.

Dokumentation zu *open*UTM

***open*UTM**
Konzepte und Funktionen
Benutzerhandbuch

Zielgruppe
Alle, die sich einen Überblick über die Funktionsbreite und Leistungsfähigkeit von *open*UTM verschaffen wollen

***open*UTM** (BS2000/OSD, UNIX, Windows)
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

Zielgruppe
Programmierer, die für die Programmierung von UTM-Anwendungen die Programmschnittstelle KDCS nutzen wollen

***open*UTM** (BS2000/OSD, UNIX, Windows)
Anwendungen generieren
Benutzerhandbuch

Zielgruppe
Anwendungsplaner, Anwendungsentwickler und Betreuer von UTM-Anwendungen

***open*UTM** (UNIX, Windows)
Einsatz von *open*UTM-Anwendungen unter UNIX und Windows
Benutzerhandbuch

Zielgruppe
Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen

***open*UTM** (BS2000/OSD, UNIX, Windows)
Anwendungen administrieren
Benutzerhandbuch

Zielgruppe
Administratoren und Programmierer von Administrationsprogrammen

***open*UTM** (BS2000/OSD)
Meldungen, Test und Diagnose
Benutzerhandbuch

Zielgruppe
Anwender, Administratoren und Programmierer von UTM-Anwendungen

***open*UTM** (UNIX, Windows)
Meldungen, Test und Diagnose
Benutzerhandbuch

Zielgruppe
Anwender, Administratoren und Programmierer von UTM-Anwendungen

***open*UTM** (BS2000/OSD, UNIX, Windows NT)
Anwendungen erstellen mit X/Open-Schnittstellen
Basishandbuch

Zielgruppe
Programmierer, die für die Programmierung von UTM-Anwendungen die X/Open-Schnittstellen CPI-C, XATMI und TX nutzen wollen

***open*UTM** (BS2000, UNIX, Windows)
***open*UTM Data Marshalling mit XML**
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Programmierer von UTM-Anwendungen

***open*UTM-Client** (UNIX)
für Trägersystem OpenCPIC
Client-Server-Kommunikation mit *open*UTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

***open*UTM-Client V5.2** (UNIX, Windows)
für Trägersystem UPIC
Client-Server-Kommunikation mit *open*UTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

***open*UTM-Client V4.0** (BS2000/OSD, UNIX, Windows)
für Trägersystem UPIC
Client-Server-Kommunikation mit *open*UTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

***open*UTM WinAdmin**
Grafischer Administrationsarbeitsplatz für *open*UTM
Online-Beschreibung (im Internet verfügbar) und Online-Hilfe

Zielgruppe
Organisatoren und Administratoren von UTM-Anwendungen

***open*UTM**, ***open*UTM-LU62** (BS2000/OSD, UNIX, Windows NT)
Verteilte Transaktionsverarbeitung
zwischen *open*UTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer und Administratoren

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-Assembler-Anwendungen

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-Fortran-Anwendungen

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-Pascal-XT-Anwendungen

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-PL/I-Anwendungen

***open*UTM**
Masterindex
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Alle, die mit der *open*UTM-Dokumentation arbeiten möchten.

Dokumentation zum openSEAS-Produktumfeld

Bean*Transactions* (Windows, UNIX)
Applikationsserver für Enterprise JavaBeans™

Administrations- und Installationshandbuch

Zielgruppe
Dieses Handbuch richtet sich an Administratoren und Designer von Bean*Transactions*-Anwendungen. Voraussetzungen sind Kenntnisse von Java und der EJB 2.0-Spezifikation von Sun Microsystems.

Bean*Transactions* (Windows, UNIX)
Applikationsserver für Enterprise JavaBeans™

Entwicklungshandbuch

Zielgruppe
Designer von Bean*Transactions*-Anwendungen und Entwickler von Enterprise JavaBeans und Java-Clients.

Biz*Transactions* V3.0
Anwendungsintegration mit Business Objekten
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Dieses Handbuch wendet sich an alle, die mit Business Objekten *open*UTM-, MVS- oder OSD-Anwendungen in Java, Windows oder openSEAS integrieren wollen.

JConnect V2.0
Verbindung von Java-Clients zu *open*UTM
Benutzerdokumentation und Java-Docs
(nur online verfügbar: Internet, *open*UTM-Handbuch-CD)

Zielgruppe
Programmierer von Java-Anwendungen

Web*Transactions*
Konzepte und Funktionen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-, MVS- oder OSD-Anwendungen ans Web anbinden wollen

Web*Transactions*
Template-Sprache
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-, MVS- oder OSD-Anwendungen ans Web anbinden wollen

Web*Transactions*
Anschluss an *open*UTM-Anwendungen über UPIC
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-Anwendungen über UPIC ans Web anbinden wollen

Web*Transactions*
Anschluss an MVS-Anwendungen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die MVS-Anwendungen ans Web anbinden wollen

Web*Transactions*
Anschluss an OSD-Anwendungen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die OSD-Anwendungen ans Web anbinden wollen

Dokumentation zum BS2000/OSD-Umfeld

AID (BS2000/OSD)
Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

Zielgruppe
Programmierer im BS2000

BCAM (BS2000/OSD)
BCAM Band 1/2
Benutzerhandbuch

Zielgruppe
Netzplaner, -generierer und -verwalter, die in BS2000-Systemen BCAM betreiben

BINDER (BS2000/OSD)
Benutzerhandbuch

Zielgruppe
Software-Entwickler

BINDER (BS2000/OSD)
Taschenbuch

Zielgruppe
Software-Entwickler, die im Umgang mit dem BINDER geübt sind

BS2000/OSD
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

Zielgruppe
BS2000/OSD-Assembler-Programmierer

BS2000/OSD-BC
BLSSERV
Bindelader-Starter
Benutzerhandbuch

Zielgruppe
Software-Entwickler und geübte BS2000/OSD-Benutzer

DCAM (BS2000/OSD, TRANSDATA)
COBOL-Aufrufe
Benutzerhandbuch

Zielgruppe
Programmierer von DCAM-COBOL-Programmen

DCAM (BS2000/OSD, TRANSDATA)
Makroaufrufe
Benutzerhandbuch

Zielgruppe
Programmierer von DCAM-Assembler-Programmen

DCAM (BS2000/OSD, TRANSDATA)
Programmschnittstellen
Beschreibung

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer, Systemverwalter und Netzadministratoren

Einführung in die XS-Programmierung
Benutzerhandbuch

Zielgruppe
Programmierer und System-Programmierer

FHS (BS2000/OSD, TRANSDATA)
Formatierungssystem für *open*UTM, TIAM, DCAM
Benutzerhandbuch

Zielgruppe
Programmierer

IFG für FHS (TRANSDATA)
Benutzerhandbuch

Zielgruppe
Datenstationsbenutzer, Anwendungsdesigner und Programmierer

FHS-DOORS (BS2000/OSD, MS-Windows)
Grafische Oberfläche für BS2000/OSD-Anwendungen
Benutzerhandbuch

Zielgruppe
End-Anwender von FHS-DOORS und Administratoren, die Masken konvertieren und diese bereitstellen

HIPLEX AF (BS2000/OSD)
Hochverfügbarkeit von Anwendungen in BS2000/OSD
Produkthandbuch

Zielgruppe
Systemverwalter und Organisatoren in Rechenzentren

IMON (BS2000/OSD)
Installationsmonitor
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an die Systembetreuung des Betriebssystems BS2000/OSD.

MT9750 V5.0 (MS Windows)
9750-Emulation unter Windows
Produkthandbuch

Zielgruppe
Das Handbuch richtet sich an den PC-Benutzer, der mit Hilfe des Produktes MT9750 mit Anwendungen auf einem BS2000-Host kommunizieren will.

OMNIS/OMNIS-MENU (TRANSDATA, BS2000)
Funktionen und Kommandos
Benutzerhandbuch

Zielgruppe
OMNIS-Administrator und OMNIS-Anwender

OMNIS/OMNIS-MENU (TRANSDATA, BS2000)
Administration und Programmierung
Benutzerhandbuch

Zielgruppe
OMNIS-Administrator und Programmierer

OMNIS-MENU (TRANSDATA, BS2000/OSD)
Benutzerhandbuch

Zielgruppe
OMNIS-MENU-Benutzer und OMNIS-MENU-Administrator

OSS (BS2000/OSD)
OSI Session Service
User Guide

Zielgruppe
OSI TP-Anwender

RSO V3.3A (BS2000/OSD)
Remote SPOOL Output
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an nichtprivilegierte Benutzer, RSO-Geräteverwalter, SPOOL-Verwalter und Systembetreuer des BS2000/OSD.

SECOS (BS2000/OSD)
Security Control System
Benutzerhandbuch

Zielgruppe
BS2000-Systemverwalter und BS2000-Anwender, die den erweiterten Zugriffsschutz für Dateien nutzen

SECOS (BS2000/OSD)
Security Control System
Tabellenheft

Zielgruppe
Erfahrene SECOS-Anwender

SESAM/SQL-Server (BS2000/OSD)
Datenbankbetrieb
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an den SESAM/SQL-Systemverwalter.

SM2 (BS2000/OSD)
Software Monitor
Band 1: Verwaltung und Bedienung

Zielgruppe
Anwender und Systembetreuung

TIAM (BS2000/OSD, TRANSDATA)
Benutzerhandbuch

Zielgruppe
nicht privilegierte BS2000-Anwender und Programmierer

VTSU (BS2000/OSD, TRANSDATA)
Virtual Terminal Support
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie System- und Netzverwalter

XHCS (BS2000/OSD)
8-bit-Code-Verarbeitung im BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie Systemverwalter, Anwender, die von EHCS auf XHCS umstellen

Dokumentation zum UNIX-Umfeld

CMX V6.0 (Solaris)
Betrieb und Administration
Benutzerhandbuch

Zielgruppe
Systemverwalter für Solaris

CMX V6.0 (Unix)
Betrieb und Administration
Benutzerhandbuch

Zielgruppe
Systemverwalter für UNIX-Systeme

CMX V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

Zielgruppe
Programmierer von Kommunikationsanwendungen auf UNIX-Systemen

OSS (SINIX)
OSI Session Service
User Guide

Zielgruppe
OSI TP-Anwender

PRIMECLUSTER*TM*
Konzept (Solaris, Linux)
Benutzerhandbuch

Zielgruppe
Benutzer und Systemadministratoren

Sonstige Literatur

BS2000/OSD Technische Beschreibung
Hrsg.: Peter Jilek
Transaktionssystem
v. Rudolf Koch
ISBN 3 89578 017 0

CPI-C (X/Open)
Distributed Transaction Processing
X/Open CAE Specification, Version 2
ISBN 1 85912 135 7

Reference Model Version 2 (X/Open)
Distributed Transaction Processing
X/Open Guide
ISBN 1 85912 019 9

TX (Transaction Demarcation) (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 094 6

XATMI (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 130 6

XML
Spezifikation des W3C (www – Konsortium)
Web page: http://www.w3.org/XML

Stichwörter

24-Bit-Modus 273
31-Bit-Modus 273

A

Abbildungsmodule, BS2000-Versionen 232
Ablauf-Koordination, Teilprogramme 36
Ablaufverfolger
  für UTM-Anwender 192
Abmelden 107
  bei Zeitüberschreitung 138
  mit KDCOFF-Kommando 138
  von UTM-Anwendung 146
Abnormal beenden
  Anwendung 98
  Neustart 95
  Teilprogramm 154
  UTM-Datenbank-Anwendung 106
Abrechnung siehe auch Accounting
Abrechnungsphase 180, 182, 186
Abrechnungssatz 181
  Aufbau 252
Abrechnungsverfahren 177
Access-List-Konzept 136
Accounting 177
  Aufbau Accountingsätze 252
  bei verteilter Verarbeitung 189
  Festpreis 181
  Rechnerleistung 177
  Satzaufbau 252
  UTM-Datenbank-Anwendung 108
  verbrauchsorientiert 181
Administrationsberechtigung 132
Administrieren
  SAT-Protokollierung 169
Adressraum, benötigter 243
Aktivieren
  STXIT-Routinen 87
ALTERNATE-LIBRARIES=NO 42
ALTERNATE-LIBRARIES=YES 42
Anlegen SYSLOG-FGG 59
Anmelden 107
  fehlerhafte Eingaben 117
  Fehlversuche 117
  ohne Benutzerkennung 132
  über OSI-TP-Clients 122
  über TS-Anwendungen 120
  über UPIC-Clients 120
  über Web 123
Anmeldeverfahren
  mit SIGNON-Services 125
Anmeldeversuche
  Fehlversuche 129
  maximale Anzahl 117
Anmeldevorgang
  Anwendungsmöglichkeiten 128
  Beispielprogramm 130, 268
  Fehlverhalten 130
  Fehlversuche 129
Anwendung
  abnormal beenden 98
  Ladestruktur optimieren 20
  mit Benutzerkennungen 135
  Speicherverbrauch 243
  Status abfragen 97
  umstellen auf BLS 55
Anwendungsbetrieb
  Vorbereitung 57
Anwendungsende
  abnormal, Neustart 95

Anwendungsglobaler Pool 28, 29
Anwendungslogik
  Teilprogramme 33
Anwendungslokaler Pool 28
Anwendungsname 113
Anwendungsprogramm
  binden (UNIX) 37
  erzeugen 33
  laden 154
  Startkommandos 89
  strukturieren 13
  umstellen auf BLS 55
Anzahl
  Asynchronvorgänge 83
  Prozesse bei Anwendungsstart 87
ASSEMB 247
ASSEMBH 247
Assembler, Version 247
Assemblieren ROOT-Tabellen-Source 33
ASSGEN 247
Asynchron-Nachricht
  ausgeben 142
Asynchron-Verarbeitung
  für UPIC-Clients 270
ASYNTASKS 83
Aufbau
  Abrechnungssatz 253
  Kalkulationssatz 254
  SAT-Protokollsatz 255
  UTM-Accountingsätze 252
Aufbereiten
  SYSLOG-Datei 71
Aufbereitungstool
  KDCCSYSL 69
  KDCPSYSL 71
Auflösen Externverweise 18
Aufrufen
  UTM-Tools 242
Auftragsschalter 97
Auftragsspezifische SAT-Protokollierung 161
Ausgabe
  Asynchron-Nachricht 142
  Basisformat 141
  KDCISAT 172
  KDCMSAT 176
  letzte Dialognachricht 144
  letzte wiederholen 145
  SM2-Messdaten 195
  Startformat 132
  wiederholen 145
Auslastung
  Betriebsmittel, kontrollieren 191
  Statistik 192
Austauschen
  Lademodul (BLS) 147
  Lademodul in CMP 151
  ONCALL-Lademodul 150
  Shared Code 26
  STARTUP-Lademodul 149
  Subsysteme 27
Ausweis 117
Ausweiskarte einlegen 117
Ausweisleser 117
Auswerten
  Messdaten (KDCMON) 200
  SM2-Messdaten 195
Auswertungslisten
  KDCMON 207
Autolink-Funktion 16, 42
  Beispiel 43
Automatisch starten nach Anwendungsende 95
Automatische Größenüberwachung
  SYSLOG 60, 65, 66
Automatischer Verbindungsaufbau 112
Automatisches KDCSIGN 120

B

Basis
  SYSLOG-FGG 62
Basisformat ausgeben 141
Basisname 57, 83
Batch-Prozesse 79
BCAM
  beenden 98
  Wartezeiten erfassen 199
BCAPPL 98

Beenden
UTM-Anwendung 97
UTM-Datenbank-Anwendung 105
UTM-Task 98
Befehlszähler 155
Beispiel
Benutzer-Protokolldatei einrichten 74
FGG auf Privatplatte einrichten 75
LLM binden mit Autolink 43
Shareable Module (OM-Format) binden 44
Beispielprogramme
Anmeldevorgang 268
Beispielprozeduren 272
Benutzer 179
Benutzerdefinierte STXIT-Routinen 155
Benutzerkennung
beim Anmelden 115
Benutzerkommandos 140
Benutzer-Protokolldatei 57, 73
doppelte 75
einrichten 73
Größe Pagepool 73
Benutzerspeicher 26, 28
Benutzerspezifisch
Meldungsmodul 38
SAT-Protokollierung 161
Berechtigungskonzept
Meldungen 137
Berechtigungsnachweis
automatischer 120
Berechtigungsprüfung 114, 133
Varianten 117
Betriebsmittel 179, 185
Auslastung kontrollieren 191
Verbrauch 108
Beweissicherung
administrieren 174
informieren über 171
SAT 157, 255
Bindelademodul 39
Bindemodul 39
Binden
Anwendungsprogramm 37
Beispiel mit Autolink-Funktion 43
Laufzeitsystem 46
UTM-Datenbank-Anwendung 104
XS-fähige UTM-Anwendung 273
BINDER 39
Anweisungen 41
BLS 147
Anwendung umstellen auf BLS 55
Programmaustausch 147
Regeln 23
BS2000
Benutzerkennung 77
Subsystem openUTM 234
BTRACE 83, 272
Bulletin 128

C

C/C++
Version 247
CATID 83
CIS 101
COB1 248, 251
COBOL
Compilerversion 248
COBOL2000 248
COBOL85 248, 251
cobrtcb2 272
Common Memory Pool
generieren 28
Shared Code 28
Compiler
unterstützte Versionen 245
Connection-Modul (DB) 104
connection-password 114
COPY-CRYPT 272
COSMOS 196
CPI-C-Beispielprogramme 270
CRTE als Subsystem laden 47

D

Darstellungsmittel 11
DATABASE 102
Dateien
notwendige für Anwendungsbetrieb 57
Dateigenerationsgruppe, siehe FGG

Datenauswertung, KDCMON 200
Datenbank 102
  Anschluss an UTM 101
  Startparameter für 88
Datenbankaufrufe erfassen 227
Datenbankschlüssel 84
Datenbanksystem
  Startparameter 84
Datenbank-Verbindungsmodul 37, 102, 104
Datenbereiche
  gemeinsam verwendbare 38
  shareable 31
Datenerfassung
  starten (KDCMON) 199
Datum
  bei Meldungen 88
DB-DIAGAREA 109
DBKEY 84
DB-System als Subsystem 104
Diagnose
  Diagnosedaten auf Datei schreiben 88
  mit SYSLOG 57
  Unterlagen bei abnormalem Beenden 99
  UTM-Datenbank-Anwendung 109
  UTM-Fehler 87
Dialognachrichten
  letzte ausgeben 144
Dienstprogramm, siehe Tools
Dokumentation, Wegweiser 3
DSSM-Anweisungen 234
  KDCMON 240
  UTM-SM2 238
DUMP 272
DUMP-CONTENT 84
Dump-Fehlercode 99
DUMP-MESSAGE 84
  Wert zurücksetzen 84
DUMP-USERID 85
Dunkelsteuerung Passwort 115
Dynamisch Programme hinzufügen 152

E

Echtzeitüberwachung SM2 193
Eingeben
  Startparameter 81
Einlegen Ausweiskarte 117
Einrichten
  Benutzer-Protokolldatei 73
Einschalten
  SAT-Protokollierung 157
  Testmodus 87
Encryption Modul 234
END 83
ENTER-JOB-Kommando 79
ENTER-Prozess 79
Entladen
  KDCMON 241
  UTM-SM2 239
  UTM-Systemcode 235
Entry-Name der Datenbank 102
Ereignisse
  sicherheitsrelevante (SAT) 158
Ereignisspezifische SAT-Protokollierung 160
Erzeugen
  Anwendungsprogramm 33
Event-Exits 17, 20, 23
Event-Services 17, 20, 23
Expliziter Verbindungsaufbau 113
Externverweise 40
  auflösen 18

F

F2/Q3-Betrieb, Sicherheitskriterien 255
Fehler
  Systemumgebung 153
Fehlermeldung
  beim Start 95
Fehlverhalten
  des Anmeldevorgangs 130
Fehlversuche
  anmelden 117
  beim Anmeldevorgang 129
Festpreis-Abrechnung 181
FGG
  auf Privatplatte einrichten 75
  PRIVATE DISK 63
  PUBLIC DISK 63

FGG
umschalten auf nächste 75
FGGUSLOG 272
FHS-DE-Formate 145
File Generation Group siehe FGG
FILEBASE 83
filebase.SLOG 58
Format
SAT-Administrationskommandos 171
Formatbibliothek 33
Formatierter Dialog 128
Formatierungssystem
Startparameter 88
Funktionstasten 117
Funktionsvarianten von UTM 142

G
Gemeinsamer Datenbereich 25
GEN 272
Generieren
shareable Module 30
UTM-Datenbank-Anschluss 102
UTM-Datenbank-Anwendung 102
Gewicht 180
festlegen 185
Grace-Sign-On 116
Größe
Pagepool (Protokolldateien) 73
Größenüberwachung
automatische 60, 65, 66
suspendierte 67

H
Hauptprogramm 36
Hinzufügen
Programme 152

I
ILCS 251
ILCS-Teilprogramm 54
ILCS-Unterstützung 54
ILCS-Version 54
Installation-Userid 235
Installieren
KDCMON 240
Produktdateien 236
UTM 231
UTM-SM2 238
INT 86

J
Jobvariable 95, 97
JOIN-Eintrag 79

K
K-/F-Taste
Benutzerkommandos 140
K000 84
K001 132
K002 112
K003 141, 143, 146
K005 137
K008 133
K009 137
K012 142
K013 141
K014 141
K015 141
K018 146
K019 146
K020 142
K021 138
K027 131
K028 119
K043 67
K049 109
K050 91
K051 91
K065 71
K068 109
K071 109
K074 150
K075 150
K078 41, 78
K079 188
K080 199
K092 119

K094 117, 129
K097 116
K120 116
K121 119
K123 137
K136 58
K138 58, 67
K146 199
K155 119
K558 71
KAA-Tabellen
dynamische 244
statische 243
Kalkulationsphase 179, 182
Kalkulationssatz 180
Aufbau 252
Kaltstart 91
KCECRYP 234
KCMSGx 69
KCNOCRYP 234
KDCADM 37
KDCCSYSL 68, 69
aufrufen 69
Meldungen 72
starten 69
KDCDIAG 199
KDCDISP 144
FHS-DE-Formate 145
KDCEVAL 201
Ergebnis 204
Meldungen 203
KDCFILE 57
Basisname 83
KDCFOR 141
KDCISAT 171
Ausgabe 172
Beispiel 172
KDCLAST 145
FHS-DE-Formate 145
KDCLOG 75
KDCMON 192, 196
Auswertungslisten 206
Daten auswerten 200
Datenerfassung starten 199
entladen 241
installieren 240
laden 241
Liefereinheiten 240
logische Maschinennummer 228
starten 197
Steuerparameter 197
UTM-Datenbank-Anwendung 108
KDCMSAT 174
Ausgabe 176
KDCOFF 146
aus Programm 138
beim Entfernen des Ausweises 117
KDCOFF BUT 146
KDCOUT 142
KDCPSYSL 68
Meldungen 72
starten 72
KDCROOT 36, 37
KDCROOT-Laufzeitsystem 38
KDCSADM 37
KDCSHUT 97
KDCSIGN 115
automatisches 120
mit Ausweis 117
KDCS-Returncode 154
KDCSWTCH 132
Kennungs-Überlaufschutz 65
Klasse-4-Speicher 26
Klasse-6-Speicher 26, 28
Kommandos
zum Starten der Anwendung 89
KTA-Trace-Area 88

L

Lademodule
Aufteilung Objekte 23
Bibliotheken 15
vorbinden 24
Lademodule (BLS)
austauschen 147
vorbinden 41
Lademodus 15

Laden
KDCMON 241
Reihenfolge Module 18
Shared Code 26
UTM-SM2 239
UTM-Systemcode 235
Ladestruktur, Empfehlungen 20
Ladevorgang 16
Laufzeitsystem 33
als Subsystem laden 47
binden 46
KDCROOT 38
Programmiersprache 38
Laufzeitsysteme, kompatible 245
LEASY 101
Leistungsengpässe analysieren 191
Leistungskontrolle 191
UTM-Datenbank-Anwendung 108
Leistungsuntersuchung, KDCMON 192
Liefereinheiten
KDCMON 240
UTM-SM2 238
LINK 272
Link Load Module 39
Linkname SYSLOG 58, 59
LLM-Format, Subsysteme 27
LLMs 39
Slices 40
vorbinden 41
LOAD-MODE=ONCALL 19
LOAD-MODE=POOL 19
Lock-/Keycode-Konzept 136
Logische Maschinennummer 228

M

Magnetstreifenkarte 117
Main Routine KDCROOT 36
Maschinennummer, logische 228
Meldungen
anmelden mit Ausweis 117
anmelden mit Benutzerkennung 114
fehlende Berechtigung 137
KDCCSYSL 72
KDCEVAL 203
KDCPSYSL 72
Meldungen mit Datum und Uhrzeit 88
Meldungsdatei
einfügen in BS2000-Meldungsdatei 236
Meldungsmodul 33
benutzerspezifischer 38
Meldungsziele
SYSLOG 57
Messdaten
auswerten (SM2) 195
erfassen (SM2) 193
Metasyntax 11
Mindestprotokollierung (SAT) 157
Module laden 18
MPOOL-Anweisung 28
MSGDMP 84
MSGMOD 272
MSGTAC 117
Multi-DB-Betrieb 101

N

Nachricht
asynchrone ausgeben 142
Namensräume
Module UTM-Systemcode 233
UTM-Systemcode 233
Nicht-priviligierte Subsysteme 26
Normales Beenden
Anwendung 97

O

Oberer Adressraum 273
Objekte
shareable 25
Objektmodule 39
OM-Format, Subsysteme 27
openUTM-Anwendung, siehe UTM-Anwendung
openUTM-Version
umstellen 251
ORACLE 101
OSI-TP-Clients
anmelden über 122
OSS-Aufrufe 86
OSS-interne Trace Records 86

OSS-Tracefunktion
ein-/ausschalten 85
OTRACE 85

P
PAMSAM 272
Parallelbetrieb
UTM 237
PASSWORD 86
Passwort
beim Anmelden 115
Zeitspanne überwachen 128
PEND ER 154, 155
Performance
Analyse über TRACE2 227
Überwachung 191
Plausibilitätsprüfung 87
Pool
anwendungsglobaler 28
anwendungslokaler 28
Postselection 168
Präfix
Startparameter 81
Preselection
SAT-Protokollierung 159
Preselection-Werte
verknüpfen 163
voreinstellen 162
PRIMARY-ALLOCATION
SYSLOG-FGG 64
PRISMA 101
Produktdateien installieren 236
Programm
dynamisch hinzufügen 152
Programmaustausch
BLS 147
ONCALL-Lademodul 150
STARTUP-Lademodul 149
Programmteile
beim Programmaufruf laden 14
beim Start laden 14
entladen 24
statisch gebundene 13
Programmübergänge 54
PROT 86
Protokollierung
aktuell eingestellte Werte 171
SAT 157, 255
Prozessornummer 113

R
RAV 181
Readme-Dateien 7
REASON 154
Recovery Phase 106
Regionsnummer 113
Registerinhalt 155
Reihenfolge
beim Laden 90
REP-Datei 233
REPFILE 233
Resource Manager
Startparameter für 88
Returncode
Abbruch 99
ROOT
Systemmodule 33, 36
Tabellen statisch binden 34
Tabellenmodul dynamisch nachladen 33
ROOTNAME 86
RTS, kompatibles 245
RUN-MODE 90

S
SAT-Mindestprotokollierung 157
SAT-Preselection 174
SAT-Protokolldatenfelder
Bedeutung 255
Versorgung 257
SAT-Protokollierung 157, 255
administrieren 169, 174
auswerten 168
benutzergesteuert 161
einschalten 157
ereignisgesteuert 160, 161
Preselection 159
Regeln 167
UTM-Datenbank-Anwendung 107

SAT-Protokollierung
Voraussetzung 170
SAT-Protokollierungswerte, informieren über 171
SAT-Protokollsatz, Aufbau 255
Schreibfehler
SYSLOG 67
SCOPE=GLOBAL 29
SECONDARY-ALLOCATION, SYSLOG-FGG 64
SERV 86
SESAM als Subsystem 104
SESAM/SQL 101
SESCOS-Auswertung 223
SET-JV-UTIL 242, 272
SFUNC 140
Shareable Datenbereich 31, 38
Shareable Module
binden (OM-Format) 44
Shareable Objekte 25
generieren 30
Shareable Programme
Common Memory Pools 14
nicht-priviligierte Subsysteme 26
Systemspeicher 13
Shareable Teile als Subsystem 47
Shareable Teilprogramme 25
Shared Code 25
austauschen 26
Common Memory Pool 28
laden 26
SHOW-ETPND 272
Sicherheitsrelevante UTM-Ereignisse 158
SIGNON-Service 125
bei UTM-Datenbank-Anwendung 107
Startformat 128
Slices
by Attributes 40
LLMs 40
SLOG-FGG 272
SM2 193
Echtzeitüberwachung 193
Liefereinheiten für UTM-SM2 238
Messdaten auswerten 195
SM2-Messdatenerfassung
Voraussetzung 193
Speicherauszug 154
Speicherplatz-Kontrolle, SYSLOG-FGG 65
Speicherverbrauch
UTM-Anwendung 243
SPI 86
Sprachanschlussmodul 38
sprachspezifische Laufzeitsysteme 33
START 83, 272
START-BLS-APPLICATION 272
Starten
KDCCSYSL 69
KDCMON 197
KDCPSYSL 72
Kommandos 89
mit WinAdmin 79
nach abnormalem Anwendungsende 95
UTM-Anwendung 77
UTM-Datenbank-Anwendung 105
UTM-Produktivanwendung 77
Startfehlercode 58 59
Startformat
Anmeldevorgang 128
ausgeben 132
START-Kommando 83
Startkommandos 83
Start-LLM 13, 19, 23, 33, 39
STARTNAME 86
Startparameter 114
Datenbank-Anwendung 88
eingeben 81
Formatierungssystem 88
Präfix 81
Syntax 82
UTM 82
UTM-Datenbank-Anwendung 105
Startprozedur 79, 92
Statistik
Anmeldeversuche 128
Auslastung 192
Statistikdaten
Auslastung 193
Struktur
Anwendungsprogramm 13
STXIT-Ereignis 155

STXIT-Routinen
aktivieren 87
benutzerdefiniert 155
Subsystem
austauschen 27
nicht-priviligiert 26
UTM 234
UTM-SM2 238
Subsystemkatalog 234
KDCMON-Einträge 240
UTM-Einträge 234
UTM-SM2-Einträge 238
SYSLIB.UTM.052.EXAMPLE 269
SYSLNK.UTM.052 37
SYSLNK.UTM.052.UTIL 242
SYSLOG 272
als einfache Datei 58
als FGG 59
filebase.SLOG 58
Größenüberwachung, automatisch 60
Link-Name 58, 59
Meldungen im virtuellen Speicher 69
Schreibfehler 67
umschalten 59, 67, 68
Verhalten bei Schreibfehlern 67
SYSLOG-Datei 57
aufbereiten 71
in SAM-Format umwandeln 69
SYSLOG-FGG 57
anlegen 61
automatische Größenüberwachung 66
Basis 62
Dateigeneration automatisch anlegen 63
Dateigenerationen zykl. überschreiben 65
Generationen erhalten 66
maximale Anzahl Dateigenerationen 61
Speicherplatz kontrollieren 65
suspendierte Größenüberwachung 67
SYSLOG-Name 58
SYSPRC.UTM.052 272
SYSPRC.UTM.052(SLOG-FGG)
SYSLOG-FGG auswerten 68
SYSPRC.UTM.052(SYSLOG)
SYSLOG auswerten 68
Systemcode
Bestandteile 232
Namensräume 233
Systemfehler 99
System-Protokolldatei 57
SYSLOG 57
Systemspeicher, Shareable Programme 13
Szenarien
UTM-Berechtigungsprüfung 117

T

Tabellenmodul KDCROOT 37
TABLIB 87
TAC-spezifische SAT-Protokollierung 161
Task
beenden 98
global auswerten mit SM2 193
starten 87
TASK PERFORMANCE TRACE 227
TASKS-IN-PGWT 87
Teilprogramm 33, 37
abnormal beenden 154
austauschen (BLS) 147
Ende erfassen (KDCMON) 228
Start erfassen (KDCMON) 227
Terminals
anmelden an openUTM über 112
TESTMODE 87
Testmodus 87
Diagnosedaten auf Datei schreiben 88
einschalten 87
TFT-Eintrag 97
Tool
KDCCSYSL 69
KDCEVAL 201
KDCPSYSL 71
TRACE2 227
Format 229
Tracefunktion BCAM ein-/ausschalten 83
Tracefunktion OSS
ein-/ausschalten 85
Trace-Records 86
TS-Anwendungen
anmelden über 120

Typ 102

U
U02 137
U16 137
Übersetzen
XS-fähiges Teilprogramm 273
Überwachung
Performance 191
UDS 105
UDS/SQL 101
Uhrzeit
bei Meldungen 88
Umschalten
auf nächste FGG 75
SYSLOG-Datei 58, 67
Umstellen
Anwendung auf BLS-Funktionen 55
Umstellung
von UTM kleiner V4.0 251
UNRESOLVED EXTERNS 40
Unterbrechungsgewicht 155
Unterbrochener Vorgang 132
Unterer Adressraum 273
UPD 272
UPIC-Clients
anmelden 120
Asynchronverarbeitung 270
UTM
installieren 231
Liefereinheiten 233
Parallelbetrieb 237
UTM-Anwendung
abnormal beenden 98
beenden 97
Speicherverbrauch 243
starten 77
Warmstart 91
UTM-Benutzerkommandos 140
UTM-Berechtigungsprüfung 133
UTM-Datenbank-Anwendung 101
abnormal beenden 106
binden 104
Diagnose 109
Leistungskontrolle 108
starten und beenden 105
UTM-Dump 99, 153
bei inkompatiblem KDCS-Returncode 85
bei K-Meldung 84
bei kompatiblem KDCS-Returncode 85
bei SIGN-Aufruf 85
in andere Kennung 85
UTM-Ereignisse
sicherheitsrelevante (SAT) 158
UTM-F-Anwendung
asynchrone Nachrichten 142
UTM-Funktionsaufrufe
erfassen 227
UTM-Meldungsdatei
einfügen in BS2000-Meldungsdatei 236
UTM-Messmonitor 192, 196
UTM-MSG-DATE 88
UTM-Report, SM2 195
UTM-S-Anwendung
ansynchrone Nachrichten 142
UTM-SAT-Administration 169
UTM-SAT-Administrationskommandos 169
Format 171
UTM-SAT-Administrator 169
UTM-SM2 193
entladen 239
laden 239
Liefereinheiten 238
UTM-STXIT 87
UTM-Systemcode
Bestandteile 232
entladen 235
laden 234, 235
Namensräume 233
UTM-Task beenden 98
UTM-Tools
aufrufen 242
UTM-Version
umstellen von 251

V
Varianten
Berechtigungsprüfung 117
Verbindung
sichern 95
Verbindungsaufbau
automatischer 112
expliziter 113
Verbindungs-Benutzerkennung 120
Verbindungskennwort 86
Verbindungsmodul 104
Datenbank 104
Formatierungssystem 37
für Datenbanken 102
Verbindungspasswort 114
Verbrauchsorientiertes Accounting 181
Verletzung der Zugriffsrechte
Meldungen 137
Verrechnungseinheiten 181
Zähler 181
Versionsnummer 16, 17
Verteilte Verarbeitung
Accounting 189
Vorbinden Lademodule (BLS) 41
Vorgang, Benutzerkennung 135
Vorgänge starten
vom OSI-TP-Client aus 134
von TS-Anwendungen aus 134
von UPIC aus 134
Vorgangswiederanlauf 135

W
Warmstart 91, 99
UTM-Datenbank-Anwendung 106
Web
anmelden über 123
WebTransactions 123
Wechselpufferverarbeitung
SYSLOG 57
Weiterschalten FGG 75
Wiederanlauf 91, 132
Wiederholen
Ausgabe 145
WinAdmin
starten mit 79
Workprozesse
Anzahl 87

X
XAP-TP System Programming Interface 86
XAP-TP-Baustein 86, 87
XML, Beispielprogramm 271
XS-Unterstützung von openUTM 273

Z
Zeitüberwachung 138
Zugangskontrolle 114
Zugriff Datenbank 84
Zurücksetzen
DUMP-MESSAGE-Wert 84
Zwischendialog
beim Anmelden 126

Inhalt

1 Einleitung 1
1.1 Konzept und Zielgruppen dieses Handbuchs 2
1.2 Wegweiser durch die Dokumentation zu *open*UTM 3
1.2.1 *open*UTM-Dokumentation 3
1.2.2 Dokumentation zum openSEAS-Produktumfeld 6
1.2.3 Readme-Dateien 7
1.3 Neuerungen in *open*UTM V5.2 8
1.3.1 Neue Plattformen 8
1.3.2 Neue Server-Funktionen 8
1.3.3 Neue Client-Funktionen 10
1.3.4 Erweiterungen für *open*UTM WinAdmin 10
1.4 Darstellungsmittel 11

2 Struktur des Anwendungsprogramms festlegen 13
2.1 Lademodule generieren 15
2.2 Module laden 18
2.3 Empfehlungen für die Strukturierung der Anwendung 20
2.4 Regeln und Einschränkungen beim Einsatz von BLS 23
2.5 Shared Code nutzen 25
2.5.1 Shared Code im Systemspeicher 26
2.5.2 Shared Code in Common Memory Pools 28
2.5.2.1 Anwendungslokaler Pool 28
2.5.2.2 Anwendungsglobaler Pool 29
2.5.2.3 Shareable Objekte generieren, die in einen Common Memory Pool geladen werden . 30

3 Anwendungsprogramm erzeugen 33
3.1 Bestandteile des Anwendungsprogramms 37
3.2 Binden mit BINDER 39
3.2.1 LLMs mit Slices 40
3.2.2 Binden von LLMs 41
3.2.3 LLMs mit Public/Private Slice binden 44
3.2.4 Laufzeitsysteme binden 46
3.2.4.1 Shareable LZS-Teile als Subsystem 47
3.2.4.2 Shareable LZS-Teile in Common Memory Pools 47
3.2.4.3 Laufzeitsysteme zu einem LLM binden 48
3.2.5 Start-LLM binden 50

3.3 Hinweise für Anwendungen mit ILCS-Teilprogrammen 54
3.4 Änderungen im Anwendungsprogramm bei der Umstellung auf die BLS-Funktionen . . 55

4 Für den Betrieb notwendige Dateien 57
4.1 System-Protokolldatei SYSLOG 57
4.1.1 SYSLOG als einfache Datei 58
4.1.2 Dateigenerationsgruppe SYSLOG-FGG 59
4.1.2.1 SYSLOG-FGG anlegen 61
4.1.2.2 Dateigeneration anlegen 63
4.1.2.3 Kennungs-Überlaufschutz 65
4.1.2.4 SYSLOG-Generationen erhalten 66
4.1.2.5 Automatische Größenüberwachung 66
4.1.3 Verhalten bei Schreibfehlern 67
4.1.4 SYSLOG-Datei auswerten 68
4.1.4.1 Das Tool KDCCSYSL - SYSLOG-Datei in SAM-Format umwandeln 69
4.1.4.2 Das Tool KDCPSYSL - Meldungstexte erzeugen 71
4.1.4.3 Meldungen von KDCCSYSL und KDCPSYSL 72
4.2 Benutzer-Protokolldatei 73
4.2.1 Benutzer-Protokolldatei einrichten 73
4.2.2 Doppelte Benutzer-Protokolldatei 75
4.2.3 Umschalten auf die nächste Dateigeneration 75
4.2.4 Verhalten bei Schreibfehlern 76

5 UTM-Anwendung starten 77
5.1 Startparameter der Anwendung 81
5.1.1 Startparameter für *open*UTM 82
5.1.2 Startparameter für das Datenbanksystem 88
5.1.3 Startparameter für das Formatierungssystem 88
5.2 Starten der Anwendung 89
5.3 Kaltstart und Warmstart 91
5.4 Beispiel einer UTM-Startprozedur 92
5.5 Fehlermeldungen beim Start 95
5.6 Nach abnormalem Anwendungsende erneut starten 95

6 UTM-Anwendung beenden 97
6.1 UTM-Anwendung normal beenden 97
6.2 UTM-Anwendung abnormal beenden 98
6.3 Diagnoseunterlagen für eine Problemmeldung 100

7 UTM-Datenbank-Anwendung 101
7.1 UTM-Datenbank-Anschluss generieren 102
7.2 UTM-Datenbank-Anwendung binden 104
7.3 UTM-Datenbank-Anwendung starten und beenden 105
7.4 Betrieb einer UTM-Datenbank-Anwendung 107
7.4.1 Anmelden und Abmelden eine Benutzers 107
7.4.2 SAT-Protokollierung 107
7.4.3 Accounting 108
7.4.4 Leistungskontrolle 108
7.4.5 Diagnose 109

8 Arbeiten mit einer UTM-Anwendung 111
8.1 Anmeldeverfahren mit Benutzerkennungen 112
8.1.1 Standard-Anmeldeverfahren für Terminals 112
8.1.1.1 Standard-Anmeldedialog 114
8.1.1.2 Automatisches KDCSIGN 120
8.1.2 Anmeldeverfahren für UPIC-Clients und TS-Anwendungen 120
8.1.3 Anmeldeverfahren für OSI-TP-Clients 122
8.1.4 Anmeldeverfahren im World Wide Web über Web*Transactions* 123
8.1.5 Mehrfach-Anmeldungen unter einer Benutzerkennung 124
8.1.6 Anmeldeverfahren mit Anmeldevorgängen 125
8.1.6.1 Anmeldevorgänge für Terminals 126
8.1.6.2 Anmeldevorgang für TS-Anwendungen 127
8.1.6.3 Anmeldevorgang für UPIC-Clients 127
8.1.6.4 Anwendungsmöglichkeiten für Anmeldevorgänge 128
8.1.6.5 Eigenschaften von Anmeldevorgängen 129
8.1.6.6 Beispielprogramme für Anmeldevorgang 130
8.1.7 Verhalten bei gesperrten Clients/LTERM-Partnern 131
8.2 Anmeldeverfahren ohne Benutzerkennungen 132
8.3 UTM-Services aufrufen 133
8.3.1 Vorgänge vom Terminal aus starten 133
8.3.2 Vorgänge vom UPIC-Client und OSI-TP-Client aus starten 134
8.3.3 Vorgänge von TS-Anwendungen aus starten 134
8.3.4 Vorgangswiederanlauf 135
8.4 Berechtigungskonzept von *open*UTM 136
8.5 Abmelden von der UTM-Anwendung 138
8.6 UTM-Benutzerkommandos für Terminals 140
KDCFOR - Basisformat ausgeben 141
KDCOUT - Asynchrone Nachricht ausgeben 142
KDCDISP - Letzte Dialognachricht ausgeben 144
KDCLAST - Letzte Ausgabe wiederholen 145
KDCOFF - Abmelden von einer UTM-Anwendung 146

9 Programmaustausch im Betrieb 147
9.1 Binden und Generieren 148
9.2 Anwendungsteile austauschen 148
9.2.1 Lademodul mit LOAD-MODE=STARTUP austauschen 149
9.2.2 Lademodul mit LOAD-MODE=ONCALL austauschen 150
9.2.3 Lademodul in einem Common Memory Pool austauschen 151
9.3 Gesamte Anwendung austauschen 151
9.4 Programme dynamisch hinzufügen 152

10 Fehlertoleranz von *open*UTM 153
10.1 Fehler, die *open*UTM erkennt 154
10.2 Fehler, die das BS2000 erkennt und zu einem STXIT-Ereignis führen 155

11 SAT-Protokollierung 157
11.1 Sicherheitsrelevante UTM-Ereignisse 158
11.2 Preselection - zu protokollierende Ereignisse einstellen 159
11.2.1 Ereignisgesteuerte SAT-Protokollierung 160
11.2.2 Benutzergesteuerte SAT-Protokollierung 161
11.2.3 Auftragsgesteuerte SAT-Protokollierung 161
11.2.4 Preselection-Werte voreinstellen 162
11.2.5 Preselection-Werte verknüpfen 163
11.3 Regeln der SAT-Protokollierung 167
11.4 Postselection - Protokollsätze auswerten 168
11.5 Administration der SAT-Protokollierung 169
11.6 UTM-SAT-Administrationskommandos 171
11.6.1 KDCISAT - Informationen über SAT-Protokollierungswerte abfragen 171
11.6.2 KDCMSAT - SAT-Protokollierung ändern 174

12 Accounting 177
12.1 Begriffsdefinitionen 179
12.2 Phasen des Accounting 182
12.2.1 Kalkulationsphase 182
12.2.2 Variante des Abrechnungsverfahrens festlegen 184
12.2.3 Abrechnungsphase 186
12.2.4 Auswertung 187
12.2.5 Fehlersituationen 188
12.3 Abrechnung bei verteilter Verarbeitung 189
12.4 Einschränkungen 190

13 Leistungskontrolle mit SM2 und KDCMON 191
13.1 Messdatenerfassung mit SM2 193
13.2 UTM-Messmonitor KDCMON 196
13.2.1 Erfassung starten und beenden 196
13.2.2 Daten auswerten 200
13.2.2.1 Daten in SAM-Format umwandeln und sortieren 200
13.2.2.2 Daten mit dem Tool KDCEVAL auswerten 201
13.2.3 Auswertungsdaten auf dem PC bearbeiten 205
13.2.4 Auswertungslisten 206
TASKS: UTILIZATION OF THE UTM TASKS 208
SUMM: TRANSACTION EVALUATION 210
TIMES: DISTRIBUTION OF PROCESSING TIMES 211
KCOP: UTM CALLS STATISTIC 212
WAIT: WAITING TIMES 215
TCLASS: EVALUATION OF THE TAC CLASSES 217
TACCL: TAC SPECIFIC TAC CLASS EVALUATION 219
TACPT: TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES 220
TACLIST: TAC SPECIFIC STATISTICS 221
TRACE: TASK SPECIFIC TRACES 222
TRACE2: TASK PERFORMANCE TRACE 227

14 Anhang 231
14.1 *open*UTM installieren 231
14.1.1 UTM-Systemcode 232
14.1.2 UTM-Systemcode laden 234
14.1.3 UTM-Systemcode entladen 235
14.1.4 Produktdateien installieren 236
14.1.5 Meldungsdateien 236
14.1.6 REP-Dateien und RMS-Dateien 236
14.1.7 Paralleler Betrieb mehrerer *open*UTM-Versionen 237
14.1.8 Subsystem UTM-SM2 238
14.1.9 Subsystem KDCMON 240
14.2 UTM-Tools aufrufen 242
14.3 Speicherverbrauch einer UTM-Anwendung 243
14.4 Compiler-Versionen, Laufzeitsysteme, KDCDEF-Optionen 245
14.4.1 Assembler 247
14.4.2 C/C++ 247
14.4.3 Cobol 248
14.4.4 Fortran 249
14.4.5 Pascal 250
14.4.6 PL/I 250
14.4.7 SPL4 250
14.4.8 Hinweise für die Umstellung von einer älteren *open*UTM-Version 251

14.5 Aufbau der Accountingsätze von *open*UTM 252
14.5.1 Aufbau des Abrechnungssatzes 253
14.5.2 Aufbau des Kalkulationssatzes 254
14.6 Aufbau der SAT-Protokollsätze 255
14.6.1 Bedeutung der von *open*UTM benutzten Protokolldatenfelder 255
14.6.2 Versorgung der Datenfelder 257
14.7 Beispielprogramme 268
14.7.1 Beispielprogramme zum Anmeldevorgang 268
14.7.2 CPI-C-Beispielprogramme 270
14.7.3 Beispielprogramme zur Asynchronverarbeitung für UPIC-Clients 270
14.7.4 Beispielprogramme für die XML-Schnittstelle 271
14.8 Beispielprozeduren 272
14.9 XS-Unterstützung von UTM-Anwendungen 273

Fachwörter 275

Abkürzungen 309

Literatur 315

Stichwörter 327

*open*UTM V5.2 (BS2000/OSD)

Einsatz von *open*UTM-Anwendungen unter BS2000/OSD

Benutzerhandbuch

Zielgruppe

Dieses Handbuch richtet sich an Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen.

Inhalt

Es enthält alle Informationen, um ein UTM-Anwendungsprogramm im BS2000/OSD zu erzeugen und eine UTM-Anwendung einzusetzen. Die Besonderheiten, die Sie beim Betrieb einer UTM-Datenbank-Anwendung beachten müssen, sind zentral in einem gleichlautenden Kapitel zusammengestellt. Ausführlich wird darauf eingegangen, wie sich Terminal-Benutzer und andere Clients an eine UTM-Anwendung anmelden können.

Zusätzlich gibt es eigene Kapitel über die Tools, die Ihnen für den Betrieb und die Kontrolle einer UTM-Produktivanwendung zur Verfügung stehen.

Ausgabe: April 2004

Datei: utm_betr.pdf

Copyright © Fujitsu Siemens Computers GmbH, 2004.

Alle Rechte vorbehalten.
Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.

Dieses Handbuch wurde erstellt von
cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Fujitsu Siemens Computers GmbH
Handbuchredaktion
81730 München

Kritik
Anregungen
Korrekturen

Fax: 0 700 / 372 00000

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Absender

Kommentar zu *open*UTM V5.2
Einsatz von *open*UTM-Anwendungen unter BS2000/OSD

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009